Ein Beitrag zur Verhaltensantizipation und -regelung kognitiver mechatronischer Systeme bei langfristiger Planung und Ausführung

Dissertation
zur Erlangung der Würde des
Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)
der Universität Paderborn

vorgelegt von
Dipl.-Inform. Philip Hartmann
33098 Paderborn

7. Dezember 2013

Dekan: Prof. Dr. Martin Schneider
Referent: Prof. Dr.-Ing. habil. Wilhelm Dangelmaier
Korreferent: Prof. Dr. rer. nat. Franz-Josef Rammig

Erstellt an der Universität Paderborn
Heinz Nixdorf Institut
Wirtschaftsinformatik, insb. CIM
Prof. Dr.-Ing. habil. Wilhelm Dangelmaier
Fürstenallee 11
33102 Paderborn

Vorwort

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Wirtschaftsinformatik, insb. CIM der Universität Paderborn innerhalb des Sonderforschungsbereichs 614 „Selbstoptimierende Systeme des Maschinenbaus“ entstanden.

In erster Linie möchte ich mich ganz besonders bei meinem Doktorvater Herrn Prof. Dr.-Ing. habil. Wilhelm Dangelmaier für die Möglichkeit zur Promotion bedanken. Während der wissenschaftlichen Betreuung meiner Arbeit hat er genau an den richtigen Stellen stets hilfreiche sowie lenkende Impulse gegeben und damit entscheidend zur Verwirklichung beigetragen. Zudem danke ich auch Prof. Dr. rer. nat. Franz-Josef Rammig für die Übernahme des Zweitgutachtens sowie den weiteren Mitgliedern meiner Prüfungskommission Prof. Dr. Leena Suhl und Prof. Dr. Stefan Betz.

Herzlich bedanken möchte ich mich auch bei Herrn Prof. Dr. Christoph Laroque und Herrn Prof. Dr. Andre Döring, die mich für eine Nebentätigkeit als IT-Berater in ihre reQuire consultants GmbH während meiner Promotion aufgenommen haben. Ihr hohes Maß an Toleranz gegenüber meiner Dissertation und das freundschaftliche Verhältnis, in dem ich arbeiten durfte, waren außergewöhnlich. Danke Jungs, ich habe sehr viel von euch gelernt!

Meinen Kollegen danke ich für die zahlreichen sowie anregenden Diskussionen und das tolle Betriebsklima an unserem Lehrstuhl. Beides hat mir auf dem Weg bis zur Fertigstellung meiner Arbeit sehr geholfen. An dieser Stelle danke ich auch allen Studierenden, die mit mir zusammengearbeitet haben, insb. Malte David Sauer, der als studentische Hilfskraft stets hervorragende Arbeit geleistet hat.

Meinen Eltern danke ich dafür, dass sie immer die Wahl meiner Ausbildung respektiert und mich dabei ermutigt sowie unterstützt haben. Besonderer Dank gilt auch meiner Freundin Jenny, die mir schon viel zu oft im Alltag den Rücken für das Schreiben meiner Dissertation freigehalten und alle meine Launen tapfer ertragen hat. Wer den Prozess einer Promotion kennt, der weiß wie unschätzbar wertvoll dies ist und was ich ihr zu verdanken habe.

Abschließend wünsche ich dem Leser auf den nun folgenden Seiten nützliche Informationen zu finden, die ihm bei der Entwicklung oder dem Betrieb von kognitiven mechatronischen Systemen weiterhelfen.

Paderborn, 7. Dezember 2013 Philip Hartmann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

> Was man heute als Science-Fiction beginnt, wird man morgen vielleicht als Reportage zu Ende schreiben müssen.
>
> (Norman Mailer)

Der Untersuchungsgegenstand dieser Arbeit ist die Antizipation und Regelung des Verhaltens von kognitiven mechatronischen Systemen, die in einer durch Nichtdeterminismus und kontinuierliche Prozesse geprägten Umwelt autonom und zielgerichtet handeln. Dabei stehen die Planung und Ausführung im Zuge eines langfristig wirtschaftlichen[1] Betriebs dieser Systeme im Vordergrund.

Kognitive Systeme können durch folgende Eigenschaften charakterisiert werden (in Anlehnung an [GRS03], S. 19):

- Einbindung und Handlung in einer Umgebung (zu der in der Regel auch andere kognitive Systeme gehören), mit der ein informationeller Austausch stattfindet
- Flexible und umgebungsadaptive Steuerung des Handelns durch Repräsentation systemrelevanter Aspekte der Umwelt
- Eine durch Lernfähigkeit und Antizipation ausgezeichnete Informationsverarbeitung

Der Begriff „*kognitives System*“ umfasst sowohl biologische als auch technische Systeme. In dieser Arbeit werden technische Systeme, deren Zusammenschlüsse und ihre optionale Interaktion mit dem Menschen fokussiert. Folglich findet ausschließlich die Betrachtung eines Teilbereiches von kognitiven Systemen statt.[2] Die eigentliche Kognition ist dabei als ein informationsverarbeitender Prozess zu ver-

[1] „*Unter Wirtschaftlichkeit im engeren Sinne versteht man das Verhältnis von Güterertrag (Nutzen) zu Produktionskosten (Aufwand). Die Wirtschaftlichkeit wird verbessert durch Kostenminimierung oder durch Steigerung des Produktionsergebnisses, z.B. über Qualitätsverbesserung oder höhere Mengen.*“ ([VDI00], S. 14, Z. 15)

[2] „*Typischerweise sind kognitive Systeme entweder Organismen (biologisch kognitive Systeme) oder technische Systeme (Roboter, Agenten). Es können aber auch Gruppen solcher Systeme (z. B. gemischte Mensch-Maschine-Verbünde) als ein (komplexes) kognitives System analysiert werden.*“ ([GRS03], S. 19, Z. 21)

stehen, der dem Wahrnehmen und Handeln zwischengelagert ist.[3]

Die Abbildung 1.1 zeigt die Grundstruktur eines mechatronischen Systems[4] mit den Elementen Grundsystem, Sensorik, Informationsverarbeitung und Aktorik.[5] Als Beispiele lassen sich Robotersysteme, elektronische Fahrzeuge, Feder-Dämpfer- und Werkzeugmaschinen-Module oder Windkraftanlagen nennen. Betrachtet man die Elemente eines mechatronischen Systems aus Sicht des Informationsflusses und hinsichtlich ihrer Funktionalität, so lässt sich die Kognition auf die Informationsverarbeitung abbilden, indem die Sensoren wahrnehmen und die Aktoren handeln.[6]

Der Forschungsbereich SFB 614[7] beispielsweise hat gezeigt, dass sich durch Integration kognitiver Komponenten mechatronische Systeme mit einer inhärenten Teilintelligenz ausstatten lassen und so ein autonomes und zielgerichtetes Handeln realisierbar ist.[8] Das Ergebnis besteht aus innovativen mechatronischen Systemen mit erweiterter Informationsverarbeitung, wie z. B. dem RailCab-System. Dabei handelt es sich um ein modulares Bahnsystem, das sich durch autonome Schienenfahrzeuge (RailCabs) für den Personen- und Gütertransport auszeichnet.[9]

[3] „*Kognitive Prozesse werden als solche der Informationsverarbeitung verstanden, also als Berechnungsvorgänge.*“ ([GRS03], S. 20, Z. 1); „*Kognition interveniert zwischen Wahrnehmen und Handeln. Sie ermöglicht die Speicherung und den Abruf früherer Erfahrungen und damit Lernen. Sie ermöglicht auch die Antizipation der Folgen eigenen Handelns und ersetzt so automatische Reiz-Reaktionskopplung durch Entscheidungen.*“ ([Gör95], S. 300, Z. 8)

[4] „*Mechatronik ist ein interdisziplinäres Gebiet der Ingenieurwissenschaften, das auf den klassischen Disziplinen Maschinenbau, Elektrotechnik und Informatik aufbaut. Ein typisches mechatronisches System nimmt Signale auf, verarbeitet sie und gibt Signale aus, die es z.B. in Kräfte und Bewegungen umsetzt.*“ (vgl. [Sch89] und [VDI04], S. 11, Z. 1)

[5] „*Beim Grundsystem handelt es sich in der Regel um eine mechanische, elektromechanische, hydraulische oder pneumatische Struktur bzw. eine Kombination aus diesen. Allgemein ist allerdings ein beliebiges physikalisches System als Grundsystem denkbar, so dass insb. auch hierarchisch strukturierte mechatronische Systeme darstellbar sind [...] Aufgabe der Sensoren ist die Bestimmung von ausgewählten Zustandsgrößen des Grundsystems. [...] Die Sensoren liefern die Eingangsgrößen für die Informationsverarbeitung [...] Die Informationsverarbeitung bestimmt die notwendigen Einwirkungen, um Zustandsgrößen des Grundsystems in gewünschter Weise zu beeinflussen. Die Umsetzung der Einwirkung erfolgt durch Aktoren direkt am Grundsystem.*“ ([VDI04], S. 14 f.); Zur detaillierten Klärung der Begriffe Grundsystem, Sensorik, Informationsverarbeitung und Aktorik sowie zur Beschreibung weiterer Beispielsysteme sei an dieser Stelle auf die weiterführende Literatur [Czi08], [Ise08] und [Rod06] verwiesen.

[6] „*Eine andere nicht direkt an die strukturelle Darstellung mit körperlichen Baugruppen (Sensor, Aktor, Mikrorechner) angelehnte Form ist die Darstellung anhand der Funktionalitäten eines mechatronischen Systems. [...] Dies ist gleichzeitig die allgemeine Darstellung einer intelligenten Maschine, die Informationen aus der Umgebung aufnimmt (Wahrnehmen), um darauf entweder umgehend zu reagieren (reaktives Verhalten), oder aufgrund eines intelligenten Erkennungsapparates (Erkennen) sinnvoll und seiner Aufgabe entsprechend zu handeln (zielorientiertes Verhalten).*“ ([Rod06], S. 23, Z. 24)

[7] Sonderforschungsbereich 614 - Selbstoptimierende Systeme des Maschinenbaus, vgl. [SFB04] und `http://www.sfb614.de/`.

[8] Vgl. [Dum10], [FGK+04], [ADG+09] und [GRS08]; siehe auch andere Projekte, wie z. B. CoTeSys - Cognition for Technical Systems, vgl. `http://www.cotesys.org/` und [BBW07].

[9] Vgl. Neue Bahntechnik Paderborn (NBP): `http://www.railcab.de/`; zur detaillierten Beschreibung der Funktionsweise vgl. auch [ADG+09].

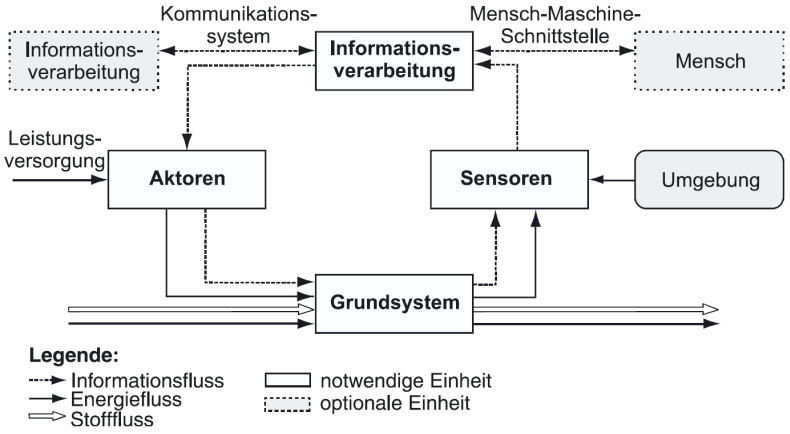


Abbildung 1.1: Grundstruktur eines mechatronischen Systems (Quelle: [VDI04])

1.1 Motivation

Die Systeme der zuvor beschriebenen Art sind u. a. in der Lage, das erforderliche Verhalten[10] ihrer Funktionsmodule zur Erfüllung eines Einzelauftrages (z. B. der Gütertransport von Station A nach Station B) eigenständig und proaktiv zu planen (Verhaltensplanung).[11] Ziel ist die Min- oder Maximierung ökonomischer Ziele, bei denen in der Regel die Minimierung des Ressourcenverbrauchs im Vordergrund steht (z. B. der Gesamtenergieverbrauch eines Einzelauftrages). Dazu wird eine Sequenz von auszuführenden Aktionen im Voraus bestimmt, die die bestmögliche Zielerreichung erlaubt (Operationsmodi des mechatronischen Systems). Die Verhaltensplanung muss dabei die kontinuierlichen Prozesse und den Nichtdeterminismus des Systemumfelds mit berücksichtigen.[12]

Weil Planung einen komplexen Problembereich bildet, und die gegebene Echtzeitanforderung[13] eines mechatronischen Systems nur eine kurze Zeit zur Durch-

[10] „*Aus Sicht der Informationsverarbeitung wird Systemverhalten durch eine Menge von Prozessen, die als eine Abfolge von Aktivitäten (activity) verstanden werden, beschrieben. Eine Aktivität ist ein konkreter, von einem Systemelement durchgeführter Prozessschritt (z.B. ‚Regelfehler bestimmen' oder ‚Sensordaten einlesen'). Im Gegensatz zum Zusammenspiel der Funktionen in einer Wirkstruktur ist die Abfolge der Aktivitäten keinen natürlichen Gesetzmäßigkeiten unterworfen und ergibt sich daher nicht rein reaktiv, sondern muss aktiv gesteuert werden. Bei einer Betrachtung auf Systemebene können verschiedene Aktivitäten einzelner Elemente zu einer Aktion (join action) zusammengefasst werden (z.B. ‚Regeln'). [...] Die Aktion ist somit eine Abstraktion, die eine Beschreibung des Gesamtvorgehens des Systems erlaubt, ohne im Detail darauf einzugehen, welche Schritte von welchen Elementen durchgeführt werden.*" ([FGK+04], S. 16, Z. 37)

[11] Zur Verhaltensplanung intelligenter mechatronischer Systeme in nicht-deterministischen Umgebungen vgl. [Klö09].

[12] Vgl. [Klö09], insb. S. 29 ff. und S. 43 ff.; vgl. auch [KAA12] und [KSWR12].

[13] „*Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.*" (Quelle: Deutsche Industrienorm (DIN) 44300)

führung eines Planungsprozesses zulässt, ist der Planungshorizont (oder die Planreichweite) zeitlich als kurzfristig einzustufen. Zusätzlich ist dieser bisher durch das Ziel des aktuellen Einzelauftrages begrenzt (z. B. das Transportfahrzeug befindet sich in Zielstation B). Es fehlen daher Informationen über mögliche Folgeaufträge, die über den begrenzten Planungshorizont hinaus an das System herangetragen werden. Folglich kann das System die Planung von Einzelaufträgen im Zuge einer weiterführenden Ausführung bzw. den erforderlichen Reaktionsspielraum im Vorfeld nicht autonom ausrichten (z. B. das Transportfahrzeug befindet sich in Zielstation B und verfügt noch über eine Energiereserve von mindestens 60%).

1.2 Ziele und Vorgehensweise

Ziel dieser Arbeit ist das Ergänzen der Informationsverarbeitung kognitiver mechatronischer Systeme um Mechanismen zur Verhaltensantizipation und -regelung.

Zunächst wird in Kapitel 2 eine detaillierte Beschreibung der Verhaltensplanung bei begrenztem Planungshorizont vorgenommen. Dabei wird ein grundlegendes Verständnis für das Verhalten, die Ziele und die Verhaltensplanung selbst entwickelt sowie insb. die aus dem begrenzten Planungshorizont resultierenden Probleme aufgezeigt. Es folgen die Betrachtung der langfristigen Verhaltensplanung und Ausführung. Ausgehend von einer hierarchischen Verhaltensplanung und -regelung werden die Verhaltensantizipation, -regelung und -planung voneinander abgrenzt und in die Hierarchie eingeordnet. Die Definition der angestrebten Wirkzusammenhänge ist dabei Gegenstand näherer Betrachtung. Es wird das Ziel verfolgt, Informationen zu charakteristischen Ausführungsverläufen zu sammeln und ein prädiktives Modell für die Verhaltensantizipation aufzubauen. Als Lösungsansatz dient das Erfassen von ausgeführten Plänen während einer rollierenden Verhaltensplanung. Anschließend werden die Anforderungen zur Realisierung des Vorhabens detailliert betrachtet und in zu lösende Teilprobleme zerlegt. Die Verhaltens- und Ereignisorientiertheit der Antizipation sowie die Klassifikation der gesammelten Ausführungsverläufe stellen hierbei wichtige Punkte dar. Danach widmet sich Kapitel 3 der Betrachtung bereits bestehender Forschungsarbeiten und etablierter Lösungen. Es wird untersucht, inwieweit diese einen Beitrag zum Lösen der Teilprobleme für das zu entwickelnde Konzept leisten können. In Kapitel 4 folgt die Präzisierung der zu leistenden Arbeit. Schließlich wird in Kapitel 5 das Erfassen und Klassifizieren von ausgeführten Plänen zum Aufbau eines um Klassen erweiterten adaptiven prädiktiven Modells konzipiert. Mithilfe gewählter Lösungsverfahren wird dann die im Zuge der ökonomischen Ziele längerfristig optimale Instruktion zur Steuerung der Verhaltensplanung bestimmt. In Kapitel 6 wird das Gesamtkonzept anhand einer Simulation des RailCab-Systems validiert. Dabei wird insb. an bestehende Arbeiten zu dem aktiven Feder-Neigesystem[14] von RailCabs angeknüpft. Abschließend fasst Kapitel 7 das Ergebnis der geleisteten Arbeit zusammen und gibt einen Ausblick auf weiteren Forschungsbedarf.

[14] Vgl. [Sch06].

2 Problemstellung

2.1 Verhaltensplanung von kognitiven mechatronischen Systemen bei begrenztem Planungshorizont

2.1.1 Verhalten

Die von einem mechatronischen System zu erfüllende Aufgabe (z. B. das Transportieren von Personen oder Gütern) lässt sich durch eine Funktion ausdrücken. Diese definiert den Zusammenhang zwischen Eingangs- und Ausgangsgrößen[15] des Systems, indem eingehende Energie-, Stoff- und Informationsflüsse in ausgehende Flüsse gleicher Art umgewandelt werden (vgl. Abb. 1.1). Teilaufgaben (wie z. B. das Fahren mit aktiver Federung) werden analog durch Teilfunktionen repräsentiert, die entsprechend logisch verknüpft die Hierarchie der Gesamtfunktion bilden.[16] Die Wirkung[17] einer Teilfunktion ist u. a. abhängig von dem gewählten physikalischen Effekt, der zu einer konkreten Teilfunktionslösung führt. Teilfunktionen können daher durch verschiedene Lösungen realisiert werden (z. B. Federung

[15] „*Die Eingangsgröße u ist eine Größe, die auf das betrachtete System einwirkt, ohne selbst von ihm [direkt] beeinflußt zu werden; sämtliche Eingangsgrößen* u_i, $i = 1, 2, 3...p$ *eines betrachteten Systems bilden den Eingangsvektor* $u = (u_1, u_2, u_3...u_p)$. *[...] Die Ausgangsgröße v ist eine erfaßbare Größe eines Systems, die nur von ihm und seinen Eingangsgrößen beeinflußt wird; sämtliche Ausgangsgrößen* $v_k, k = 1, 2, 3...q$ *eines betrachteten Systems bilden den Ausgangsvektor* $v = (v_1, v_2, v_3, ...v_q)$.“ ([DIN94], S. 3, Z. 52)

[16] „*Zum Beschreiben und Lösen konstruktiver Aufgaben ist es zweckmäßig, unter Funktion den gewollten Zusammenhang zwischen Eingang und Ausgang eines Systems mit dem Ziel, eine Aufgabe zu erfüllen, zu verstehen. [...] Eine Gesamtfunktion lässt sich in vielen Fällen sogleich in erkennbare Teilfunktionen aufgliedern, der dann Teilaufgaben innerhalb der Gesamtaufgabe entsprechen. Die Verknüpfung der Teilfunktionen zur Gesamtfunktion unterliegt dabei sehr häufig einer gewissen Zwangsläufigkeit, weil bestimmte Teilfunktionen erst erfüllt sein müssen, bevor andere sinnvoll eingesetzt werden können. [...] Die Funktionen werden [...] mit einem Haupt- und Zeitwort wie ‚Druck erhöhen‘, ‚Drehmoment leiten‘, ‚Drehzahl verkleinern‘ beschrieben.*“ ([PBFG05], S. 42, Z. 9)

[17] „*Wirkung [...] ist die Beeinflussung einer Größe, der beeinflußten Größe, durch eine oder mehrere andere Größen, die verursachenden Größen. [...] In der Regelungs- und Steuerungstechnik besteht zwischen der beeinflußten Größe und den sie verursachenden Größen ein funktionaler Zusammenhang, der die wirkungsmäßige Abhängigkeit beschreibt und beispielsweise durch eine Systemanalyse zu finden ist.*“ ([DIN94], S. 3, Z. 68)

mit hoher oder niedriger Störgrößenkompensation).[18] Die Wahl der Teilfunktionslösungen bestimmt die Lösung der Gesamtfunktion und damit die Wirkung des mechatronischen Gesamtsystems.

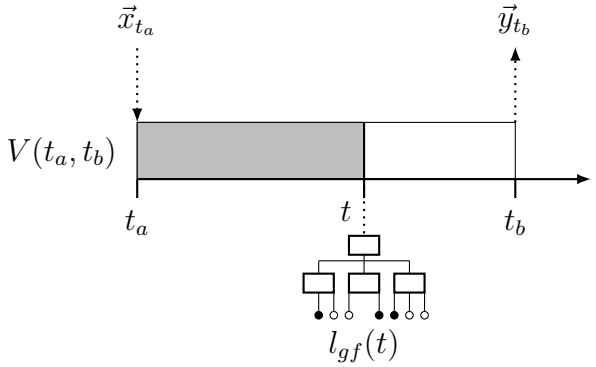


Abbildung 2.1: Verhalten eines mechatronischen Systems im Zeitverlauf

Darunter ist Folgendes zu verstehen: Es seien TF_{leaf} die Menge aller Teilfunktionen der untersten Hierarchieebene innerhalb der Gesamtfunktion[19] und L_{tf} die Menge aller Lösungen von Teilfunktion tf, die das mechatronische System implementiert[20], dann können die gewählten Lösungen (vgl. Abb. 2.1, schwarze Kreise) innerhalb der Gesamtfunktion zu einem Zeitpunkt t als eine Folge von Teilfunktionslösungen aufgelistet werden: $l_{gf}(t) = (l_{tf_1}, ..., l_{tf_k})$ mit $l_{tf_i} \in L_{tf_i}$ und $1 \leq i \leq k = |TF_{leaf}|$. Damit ist das Verhalten des Gesamtsystems zu einem Zeitpunkt t beschreibbar als $V(t) = (\vec{x}_t, l_{gf}(t), \vec{y}_t)$ mit $\vec{x}_t$ als Eingangsgrößenvektor und $\vec{y}_t$ als Ausgangsgrößenvektor. Folglich ergibt sich für das Verhalten in einem Zeitraum: $V(t_a, t_b) = (V(t_a), ..., V(t_b))$ mit $t_a \leq t_b$ (vgl. Abb. 2.1).[21] Die Wirkung einzelner Teilfunktionslösungen ist dabei abhängig von der Verknüpfung der zugehörigen Teilfunktion mit anderen Teilfunktionen und dazu gewählten Teilfunktionslösungen.

Das mechatronische System reagiert im Zeitverlauf auf die jeweils anliegende Eingangsgröße, indem es diese durch die momentan realisierte Gesamtfunktion (Verknüpfung aktueller Teilfunktionslösungen gemäß der Funktionshierarchie) in die

[18] „*Teilfunktionen werden in der Regel durch physikalisches, biologisches Geschehen erfüllt, wobei das erstere in maschinenbaulichen Lösungen überwiegt [...] Der Wirkzusammenhang wird daher von den gewählten physikalischen Effekten und den festgelegten geometrischen und stofflichen Merkmalen bestimmt [...] Eine Teilfunktion kann oft von verschiedenen physikalischen Effekten erfüllt werden [...] Der gewählte physikalische Effekt einer Teilfunktion muss aber mit den Effekten von anderen verknüpften Teilfunktionen verträglich sein.*“ ([PBFG05], S. 50, Z. 3)

[19] „*Theoretisch lassen sich Funktionen so aufgliedern, dass die unterste Ebene der Funktionsstruktur nur aus Funktionen besteht, die sich hinsichtlich allgemeiner Anwendbarkeit praktisch nicht weiter unterteilen lassen.*“ ([PBFG05], S. 45, Z. 10)

[20] Vgl. hierzu auch [Klö09], S. 7.

[21] „*Das Ein- und Ausgabeverhalten des Gesamtsystems lässt sich als Tupel aus Eingabewerten aus dem Umfeld, einer Folge von Teilfunktionslösungen und Ausgabewerten beschreiben.*“ ([Klö09], S. 7, Z. 20); vgl. auch [PBFG05].

Ausgangsgrößen umwandelt (reaktives Verhalten). Weil insb. die zwischengelagerte Kognition bzw. Informationsverarbeitung eine gewisse Zeit in Anspruch nimmt, kommt es zu Latenzzeiten bzgl. der Reaktion auf beteiligte Eingangsgrößen. Daher verzögert sich die Wirkung der Gesamtfunktion auf bestimmte Ausgangsgrößen. Da das System die Berechnung möglichst echtzeitnah durchführen soll, ist dies bei der Entwicklung von kognitiven Verfahren mitzuberücksichtigen.

Eine Möglichkeit dieses Problem zu lösen, ist z. B. das Verwenden einer Systemarchitektur in Form des Operator-Controller-Moduls (OCM) nach Naumann[22]. Das OCM gliedert sich in drei Systemebenen: Kognitiver Operator, reflektorischer Operator und Controller mit jeweils angrenzenden Regelkreisen. Der Controller (unterste Ebene) verarbeitet (quasi-)kontinuierlich die Messsignale der Sensorik und gibt Stellsignale für die Aktorik aus. Der reflektorische Operator (mittlere Ebene) überwacht den Controller und steuert diesen durch Initiieren von Parameter- oder Strukturänderungen. Im kognitiven Operator (oberste Ebene) findet das Anwenden von Lern-, Optimierungs- oder Planungsverfahren statt. Der reflektorische Operator definiert damit einen Übergang von einer Handlungsebene unter harter Echtzeit zu einer Planungsebene unter weicher Echtzeit.[23] Die in dieser Arbeit beschriebenen bzw. entwickelten Methoden sind demzufolge ausschließlich dem kognitiven Operator zuzuordnen.

Zur autonomen Steuerung des Verhaltens sind daher von dem System eigenständig Teilfunktionslösungen auszuwählen und zu bestimmten Zeiten zu aktivieren, um die entsprechende Wirkung zu erzielen (aktives Verhalten). Die Auswahl erfolgt dabei durch einen Abgleich mit konkreten Zielsetzungen. Genau darin besteht das zu lösende Entscheidungsproblem des kognitiven mechatronischen Systems: Teilfunktionslösungen auszuwählen, die das bestmögliche Erreichen dieser Ziele mit hoher Sicherheit zur Folge haben (zielgerichtetes Verhalten).[24]

2.1.2 Ziele und Zielsysteme

Kognitive mechatronische Systeme, die autonom handeln, verfolgen eigenständig Ziele.[25] In diesem Zusammenhang ist zwischen Zielfunktionen (engl. *objectives*) und Zielzuständen (engl. *goals or goal states*) zu unterscheiden. Zielfunktionen geben eine Richtung an, in der sich Werte bis hin zu einem Optimum verbessern (z. B. minimiere Energieverbrauch), während Zielzustände konkrete Zustände repräsentieren, in denen sich das System nach Abschluss eines Einzelauftrages befinden

[22] Vgl. [Nau00].

[23] Vgl. hierzu auch [ADG+09], S. 13 ff.

[24] „*The major requirement for autonomy is that the entity must be trying to accomplish something; that is, it must be goal directed.*“ ([CL91], S. 111, Z. 83)

[25] „*Ein Ziel ist ein als möglich vorgestellter Sachverhalt, dessen Verwirklichung erstrebt wird; es wird durch eine Entscheidung gesetzt. Sachverhalte sind z.B. Zustände, Gegenstände, Handlungen, Prozesse, Beziehungen. [...] Ein Ziel wird in einem Zielsatz formuliert. Ein Zielsatz enthält zwei Bestandteile: (a) die beschreibende Kennzeichnung des Sachverhaltes, (b) die Auszeichnung dieses Sachverhaltes als erstrebt, erwünscht, gefordert, befürwortet.*“ ([VDI00], S. 4, Z. 8 und vgl. auch [Rop79], S. 116)

soll (z. B. das Transportfahrzeug ist in Station B).[26] Zusammen definieren diese dann die Menge an optimalen bzw. gültigen Zielzuständen für den aktuellen Einzelauftrag des kognitiven mechatronischen Systems (z. B. das Transportfahrzeug ist in Station B und verbrauchte das mögliche Minimum an Energie). Teilfunktionslösungen und damit erreichte Teilziele sind das Mittel[27] zur Umsetzung dieser Ziele.

Kognitive mechatronische Systeme verfolgen in der Regel nicht nur ein Ziel, sondern mehrere und oft sogar gegensätzliche Ziele (engl. *multi objective*, z. B. im Fall des aktiven Feder-Neigesystems eines RailCabs: Minimierung des Energieverbrauchs und Maximierung des Komforts, wobei die Erhöhung des Komforts mit einer Erhöhung des Energieverbrauchs einhergeht). Da Mehrzieloptimierung (auch als Pareto-Optimierung bezeichnet) komplex ist, wird in der Regel die Menge an optimalen Teilfunktionslösungen (auch als Pareto-Menge bezeichnet) für typische Szenarien zur Entwurfszeit vorbereitend bestimmt und diese in der Betriebszeit für entsprechend klassifizierte Situationen zur Auswahl bereitgestellt (Operationsmodi).[28]

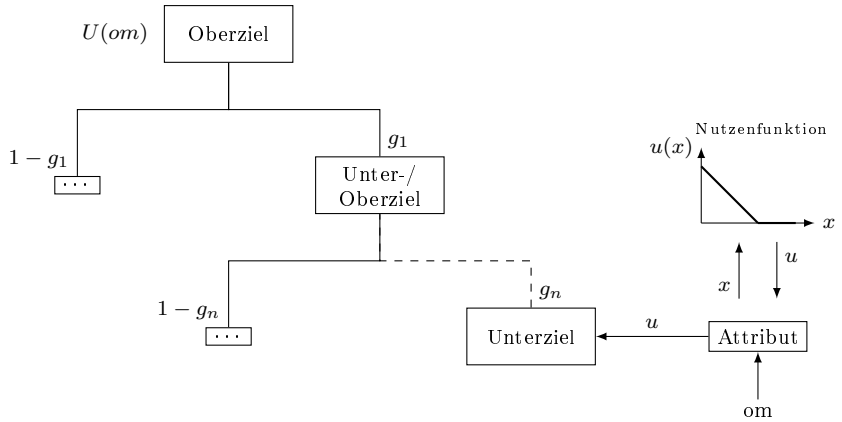


Abbildung 2.2: Zielsystem eines kognitiven mechatronischen Systems

[26] „*An objective generally indicates the ‚direction' in which we should strive to do better [...] A goal is different from an objective in that it is either achieved or not.*" ([KR93], S. 34, Z. 5)

[27] „*Ein Mittel dient dazu, ein Ziel zu erreichen. Man spricht dann von einer Instrumentalbeziehung. Jedes Mittel kann selbst wiederum als Ziel betrachtet werden. Häufig gilt auch die Umkehrung, dass ein Ziel als Mittel zur Verwirklichung eines anderen Zieles anzusehen ist. Die Kenntnis und die Gestaltung von Mitteln können rückwirkend auch ein Ziel verändern.*" ([VDI00], S. 5, Z. 14)

[28] Vgl. [GWTD08], [VT08], [Klö09] und [KAA12].

Bestehen zwischen den Zielen zusätzlich Relationen, so kann von einem Zielsystem[29] (engl. *system of objectives or goals*) gesprochen werden (vgl. Abb. 2.2).[30] Ziele eines Zielsystems werden entweder direkt zur Entwurfszeit festgelegt (engl. *built-in goals or objectives*)[31] oder während der Betriebszeit erworben (engl. *acquired goals or objectives*). Dazu kann das kognitive mechatronische System eigenständig Ziele generieren (endogene oder interne Ziele) oder Ziele aus seinem Umfeld aufnehmen (exogene oder externe Ziele).[32] Insgesamt besteht nun das kognitive mechatronische System aus drei Hauptkomponenten: Zielsystem, Informationsverarbeitung und Grundsystem (vgl. auch Abb. 1.1). Dies entspricht im Wesentlichen der Grobstruktur eines Handlungssystems nach Ropohl[33], bei der die Informationsverarbeitung mit dem in dieser Arbeit bereits erwähnten OCM gleichzusetzen ist.

Die Informationsverarbeitung muss während der Betriebszeit autonom entscheiden, welcher Operationsmodus für die aktuelle Situation bzgl. des Zielsystems optimal ist. Eine Präferenz dieser Entscheidungsalternativen ist z. B. durch eine Kombination aus Entscheidungsbaumanalyse[34] und Nutzentheorie[35] aufstellbar. Hierbei wird das Zielsystem als ein Entscheidungsbaum mit Ober- und Unterzielen aufgebaut. Kanten zu Ober- oder Unterzielen werden mit Gewichten $g \in [0, 1]$ versehen, die angeben, wie stark die untergeordneten Ziele in die Zielerfüllung des übergeordneten Ziels mit einfließen. Dabei gilt für alle Gewichte einer Hierarchieebene des Baumes $\sum g = 1$ (vgl. Abb. 2.2).

[29] „*Ein System heißt Zielsystem S_Z, wenn in $S_Z = (\sigma, \pi)$ $\sigma = \{Z_K\}$ mit $k \in K \sim \mathbb{N}$ eine Menge von Zielen ist und $\pi = \{P_{Idfm}\} \cup \{P_{Kkrm}\} \cup \{P_{Ismm}\} \cup \{P_{Prfm}\}$ eine Menge von Indifferenz-, Konkurrenz-, Instrumental- und Präferenzrelationen ist.*“ ([Rop79], S. 117, Z. 8); Zur Klärung der Relationenbegriffe vgl. [Rop79], S. 117 ff.

[30] „*Ein Ziel ist häufig Bestandteil eines Zielsystems, das mehrere Ziele und Beziehungen zwischen den Zielen umfasst. Ist ein bestimmtes Ziel in einem allgemeineren Oberziel enthalten oder enthält es selbst speziellere Unterziele, so liegt eine begriffliche Hierarchiebeziehung vor. Durch die Angabe von Unterzielen kann konkretisiert werden, was mit einem Ziel genau gemeint ist.*“ ([VDI00], S. 4, Z. 29)

[31] Zum Entwurf von Zielsystemen mechatronischer Systeme vgl. u. a. [Poo11].

[32] „*Built-in goals exist [...] when the system is ‚born‘ and starts its activities. [...]. Acquired goals are created in a system after the system starts its activities. They are not necessarily subgoals of existing goals within the system in a problem-solving process, yet they are non-existent when the system starts its ‚life‘. [...] Endogenous goals are created by and within the system. These goals might be created as a reaction to some stimulus from the environment or as subgoals in the process of problem solving. Exogenous goals are created outside the system and become its goals either at the time the system was designed or through its sensors, but in either case, these are already formulated as goals. The distinction between endogenous and acquired goals is that endogenous goals are created exclusively through the intelligent mechanisms of the system; that is, the goals are created through learning, problem solving, or interaction with an external environment but not through direct programming. Exogenous goals include the set of built-in goals and also goals that are set by an external agent to the system, for example, by a direct command.*“ ([CL91], S. 114, Z. 70); vgl. auch inhärente, interne und externe Ziele in [FGK+04], S. 30 f.

[33] Ropohl strukturiert ein Handlungssystem in Zielsetzungs-, Informations- und Ausführungssystem (vgl. [Rop09], S. 89 ff. und [Rop79]).

[34] Vgl. z. B. [KR76].

[35] Vgl. z. B. [Fre86].

Für die Unterziele werden Attribute definiert, die zu einem Operationsmodus Eigenschaften festlegen, die in die Beurteilung der Zielerfüllung mit einfließen (z. B. Energieverbrauch). Die Auswertung eines Attributs liefert dann einen konkreten Eigenschaftswert bzw. Effekt x des Operationsmodus (z. B. Energieverbrauch in Wattsekunden). Um die Eigenschaftswerte verschiedener Attribute zu normieren, werden zusätzlich Nutzenfunktionen definiert, die zu einem Eigenschaftswert den Nutzen $u(x) \in [0,1]$ angeben. Diese sind zur Entwurfszeit von einem Entwickler definiert worden und können ggf. zur Betriebszeit modifiziert werden. Zu jedem Operationsmodus om ist dann der Gesamtnutzen $U(om) \in [0,1]$ an der Wurzel des Zielsystems (die Zielerfüllung des Oberziels) durch Aufwärtspropagieren der Nutzenwerte durch den Baum unter Einberechnung der Gewichtungswerte bestimmbar. Mithilfe dieser Metrik können verschiedene Operationsmodi verglichen und für die aktuelle Situation eine Wahl getroffen werden (vgl. Abb. 2.2). Dabei ist das Zielsystem z. B. durch externe Eingriffe eines Benutzers indirekt mittels der Kantengewichte zur Betriebszeit beeinflussbar.[36]

Damit ergibt sich die Planungsaufgabe für das kognitive mechatronische System: Proaktiv Sequenzen von Operationsmodi festlegen, die in Summe den bestmöglichen Nutzenwert des Zielsystems erreichen.[37]

2.1.3 Verhaltensplanung

Um die im Sinne des gegebenen Zielsystems (vgl. Abs. 2.1.2) und zum Erreichen der Auftragserfüllung bestmögliche Sequenz von Operationsmodi proaktiv zu bestimmen, ist im Vorfeld ein Planungsprozess durchzuführen. Dieser wird als Verhaltensplanung bezeichnet und verlangt als Eingabe, neben dem aktuellen Systemzustand[38] und den Zielvorgaben, ein Planungsmodell zu dem Verhalten (vgl. Abs. 2.1.1) des mechatronischen Systems. Klöpper[39] unterscheidet bspw. hierbei zwischen einem deterministischen Planungsmodell (vgl. Abs. 2.1.3.1) und einem darauf aufbauenden probabilistischen Planungsmodell (vgl. Abs. 2.1.3.2). Um die damit verbundenen Probleme hinsichtlich eines längerfristig ökonomischen Betriebs aufzuzeigen, sollen diese Planungsmodelle im Folgenden detaillierter betrachtet werden.

2.1.3.1 Deterministisches Planungsmodell

Als Erstes wird das deterministische Planungsmodell zum Verhalten des mechatronischen Systems untersucht, weil dieses die Grundlage für die spätere probabilistische Modellerweiterung bildet.

[36] Vgl. [KRS+08] und [KRSV08].

[37] Vgl. [Klö09], [KAA12] und [KSWR12].

[38] „*Zustandsgrößen $x_j, j = 1,2,3...n$ sind diejenigen zeitveränderlichen Größen eines Systems, mit deren Kenntnis zu irgendeinem Zeitpunkt das weitere Verhalten des Systems bei gegebenen Eingangsgrößen eindeutig bestimmbar ist. Die Gesamtheit der Zustandsgrößen des betrachteten Systems bilden den Zustandsvektor $x = (x_1, x_2, x_3...x_n)$.*“ ([DIN94], S. 3, Z. 11)

[39] Vgl. [Klö09].

Klöpper et al.[40] formulieren ein deterministisches Planungsmodell für kognitive mechatronische Systeme wie folgt:[41]

- OM ist eine endliche Menge von verfügbaren Operationsmodi
- S ist eine endliche Menge von möglichen Systemzuständen
- $\vec{s} \in S$ ist Zustandsvektor mit $s(i) \in \mathbb{R}$ als i-te Komponente

Weiterhin bezeichnet für jeden Operationsmodus $om \in OM$:

- $prec^{om} := \{(x_{lower} < s(i) < x_{upper}) | x_{lower}, x_{upper} \in \mathbb{R}\}$ die Menge von Vorbedingungen, die zur Ausführung von om zu erfüllen sind
- $post^{om}$ eine Menge bedingter numerischer Funktionen zur Beschreibung von Effekten auf Zustandsvariablen im Folgezustand s'

Die Ausprägung in Form eines konkreten Planungsproblems besteht dann in der Suche nach einer Folge von Operationsmodi, die ausgehend von einem initialen Systemzustand $\vec{s}_i \in S$ einen Übergang zu einem vorher festgelegten Zielzustand $\vec{s}_g \in S$ festlegt. Einzelaufträge (engl. *orders*) des mechatronischen Systems lassen sich somit als 2-Tupel $O = (\vec{s}_i, \vec{s}_g)$ beschreiben. Im einfachsten Fall kann das jeweilige deterministische Planungsproblem z. B. mittels einer zustandsbasierten Suche[42] (engl. *state space search*) gelöst werden. Das Finden der optimalen Sequenz von Operationsmodi $om_1, ..., om_n$ erfolgt dann durch das Abgleichen des Gesamtnutzenwertes $\sum_{i=1}^{n} U(om_i)$ abhängig von dem gegebenen Zielsystem (vgl. Abs. 2.1.2).[43]

2.1.3.2 Probabilistisches Planungsmodell

Da der Nichtdeterminismus der Systemumgebung, z. B. verursacht durch nicht exakt bestimmbare Wetterverhältnisse, zum Berechnen der bestmöglichen Lösung während der Verhaltensplanung miteinzubeziehen ist (Planung unter Unsicherheit), muss das zugrundeliegende deterministische Planungsmodell (vgl. Abs. 2.1.3.1), um diesen abzubilden, entsprechend erweitert werden.

Nach Klöpper et al.[44] besteht ein probabilistisches Planungsmodell für kognitive mechatronische Systeme zum einen aus probabilistischen Zuständen s^p mit

$$range(s^p(i)) \rightarrow W(\mathbb{R})$$

[40] Vgl. [KAA12].

[41] „*Adaptive mechatronic systems are able to execute a function in different ways. From the planning perspective, these different ways are distinguished from each other by how well they achieve the possible objectives and how they change the system state. We refer to these different implementations of functions as operation modes [...] In a mechatronic planning domain most relevant variables are numerical, hence we restrict the state vector to real valued numerical variables [...] The transition function of an operation mode is a set of conditional numerical functions describing the change of influenced state variables. A condition is a logical expression (conjunctions and disjuctions) of comparison operations. If a condition is true, the result of the corresponding numerical function is assigned to state variable [...] in the next state of the plan.*“ ([KAA12], S. 175, Z. 86)

[42] Vgl. State-Space Planning [GNT04], S. 69 ff.

[43] Vgl. [KAA12].

[44] Vgl. [KAA12].

für den Wertebereich (z. B. $0 \leq SOC_k \leq 100$ mit SOC_k für die Energieverfügbarkeit (*state of charge*) des mechatronischen Systems in % im Zustand k) und

$$distribution(s^p(i))$$

für die Wahrscheinlichkeitsverteilung zur Zustandsvariablen $s^p(i)$ (z. B. $\mathbb{P}(SOC_k \leq 50) = 0.25 \wedge \mathbb{P}(SOC_k > 50) = 0.75$) und zum anderen aus probabilistischen Varianten der Operationsmodi des mechatronischen Systems mit

- $in_s^{om} \subseteq pre^{om}$ für eine Teilmenge an Eingangsvariablen und
- $out_s^{om} \subseteq post^{om}$ für eine Teilmenge an Ausgangsvariablen

des Operationsmodus om.

Zu jeder Ausgangsvariablen $o \in out_s^{om}$ wird ein Bayes'sches Netz[45] bn_o^{om} aufgestellt und so der probabilistische Effekt Δ auf der Zustandsvariablen eines Folgezustands $k+1$ mittels bedingter Wahrscheinlichkeiten formuliert (z. B. $\mathbb{P}(SOC_{k+1}^{\Delta} = +10 | WindKmh \leq 20 \wedge SOC_k > 50) = 0.015$). Mit der Angabe einer unteren bzw. oberen Schranke für kritische Zustandswerte (z. B. $SOC_{k+1} \leq 10$) können Verzweigungsstellen, in denen die Verletzung dieser Schranke am wahrscheinlichsten ist, bestimmt und dafür alternative Planungsverläufe generiert werden.[46]

Das Problem beim Verwenden eines derartigen probabilistischen Modells besteht darin, dass zum einen die Wahrscheinlichkeitsverteilungen zu den Umgebungseinflüssen vorab bekannt sein müssen und zum anderen diese dann im Allgemeinen fixiert werden. Da aber während des Betriebs eines mechatronischen Systems innerhalb eines längeren Zeitraums das Vorkommen von Umgebungseinflüssen grundlegend variieren kann, ist daher die Adaptivität eines solchen Planungsmodells zu gewährleisten.

2.1.4 Problem des begrenzten Planungshorizontes

Bisherige Konzepte zur Verhaltensplanung (vgl. Abs. 2.1.3) erlauben eine Beurteilung zukünftiger Auswirkungen, solange die externen Ziele (vgl. Abs. 2.1.2) sowie die Systemumgebung bekannt sind. In der Realität findet jedoch bei den hier betrachteten Systemen eine fortwährende Konfrontation mit einem offenen Planungshorizont statt. Es wird ein Zeithorizont betrachtet, für den beispielsweise aufgrund fehlender Informationen zu Folgeaufträgen die externen Zielsetzungen und damit die zukünftigen Systemumgebungen noch unbekannt sind. Die für den begrenzten und kurzfristigen Planungshorizont von der Verhaltensplanung eingeplanten Operationsmodi sind daher bzgl. weiterführender Folgeaufträge in der Gesamtbetrachtung ggf. nicht optimal. Folglich ist die Verhaltensplanung der Einzelaufträge in geeigneter Weise zu beeinflussen. Es muss sichergestellt werden, dass trotz sich verändernder externer Ziele sowie Systemumgebung langfristig bestmögliche Ergebnisse erreichbar bleiben.

Anhand von historischen Daten sollten daher mögliche zukünftige Verläufe, bestehend aus voraussichtlichen Systemzuständen und Aktionen, bestimmt und in den

[45] Vgl. [BG07].
[46] Vgl. [Klö09], [KSWR12] und [KAA12].

aktuellen Entscheidungsprozess eingebunden werden. Dabei können die Darstellungen der zukünftigen Verläufe nicht in der Detaillierung erfolgen, die der Verhaltensplanung zu Grunde liegt. Zudem kann nicht immer davon ausgegangen werden, dass die Planungsdomäne derart gut strukturiert ist, sodass ein detailliertes probabilistisches Modell auf Basis von Bayes'schen Netzen (vgl. Abs. 2.1.3.2) formulierbar ist. Aus diesen Gründen sind zusätzliche Komponenten der Informationsverarbeitung zu entwickeln, die anhand der bisher aufgetretenen Ausführungssequenzen vereinfachte charakteristische Verläufe generieren. Um als Entscheidungsgrundlage dienen zu können, sind diese Verläufe mit der aktuellen Situation in Beziehung zu setzen. Idealerweise werden die möglichen zukünftigen Verläufe entsprechend gewichtet. Die Bestimmung dieser Gewichtung sollte dabei alle verfügbaren Informationen nutzen. Auf Basis der bestimmten und hinsichtlich ihrer Eintrittsmöglichkeiten bewerteten charakteristischen zukünftigen Verläufe können dann Konsequenzen für die aktuellen Entscheidungsprozesse abgeleitet werden. So wäre beispielsweise eine Präzisierung der Aktionen denkbar, die zur Erfüllung eines Einzelauftrages während der Verhaltensplanung (vgl. Abs. 2.1.3) eingeplant sind. Auf diese Weise könnten mögliche zukünftige externe Ziele und Umwelteinflüsse mit der kurzfristigen Verhaltensplanung abgeglichen werden.

Die Begrenzung des Planungshorizontes der Verhaltensplanung und der genannte Ansatz zur Lösung dieses Problems verlangen nach einer übergeordneten Ebene zur langfristigen Planung. Im Zuge eines langfristig ökonomischen Betriebs von kognitiven mechatronischen Systemen muss ein Übergang vom kürzeren Planungshorizont der Verhaltensplanung zum längeren Planungshorizont der übergeordneten Ebene geschaffen werden.

2.2 Langfristige Verhaltensplanung und Ausführung von kognitiven mechatronischen Systemen

Ein kognitives mechatronisches System, das sein Verhalten eigenständig im Voraus plant, sollte diesen Prozess mit unterschiedlichen zeitlichen sowie inhaltlichen Bezügen durchführen können (Reichweite der Planung). Insgesamt ist von einer Hierarchie mit mehreren Ebenen der Verhaltensplanung auszugehen.[47]

In dieser Arbeit wird, repräsentativ für diese Hierarchie, ausschließlich zwischen zwei Planreichweiten unterschieden: Der zeitlich kürzeren und inhaltlich detaillierenden Planreichweite einer unteren Ebene (z. B. die Planung der Operationsmodi für einen Streckenabschnitt) und der zeitlich längeren und inhaltlich vergröbern-

[47] „*Im Hinblick auf die zeitliche und inhaltliche Reichweite von Entscheidungen unterscheidet man drei Hauptebenen der Planung, die im Sinne einer sukzessiven Verfeinerung in einem hierarchischen Zusammenhang stehen: • strategische (langfristige) Planung • taktische (mittelfristige) Planung • operative (kurzfristige) Planung.*“ ([Sch01], S. 14, Z. 35); vgl. auch Long-term, Mid-term und Short-term planning in [FMW05].

den Planungsreichweite einer überlagernden oberen Ebene (z. B. die Planung einer Fahrtroute).[48] Sind bei der Verhaltensplanung einer Ebene der Planungshorizont und Planungszyklus (oder auch Planabstand, zeitlicher Abstand zwischen zwei Planungsprozessen) derart gewählt, dass der Zeitraum zwischen aktuellem Zeitpunkt und dem Planungshorizont größer als der Planungszyklus ist, dann kann das gesamte Vorgehen während der Verhaltensplanung als rollierende Planung[49] aufgefasst werden.[50]

Die Ausführung des mechatronischen Systems erfolgt verzahnt mit einer Ausführungsüberwachung. Diese überprüft nach jeder aktuellen ausgeführten Aktion, ob für den erreichten Systemzustand eine eingeplante Folgeaktion existiert, oder eben eine Neu- bzw. Umplanung erforderlich ist.[51] Dadurch entsteht die Möglichkeit zum Aufbau eines Modells innerhalb der überlagernden Ebene mit Ergebnissen aus der Planausführung der angrenzenden überlagerten Ebene. Auf Basis dieses Modells kann dann das längerfristige Verhalten der überlagerten Ebene antizipiert werden und die Informationen dazu mit in die Verhaltensplanung der überlagernden Ebene einfließen.[52] Zudem sind mit der gegebenen Rückführung von überlagerter zur überlagernder Ebene, und fortwährender Anpassung des Verhaltens an die Ziele des Zielsystems (vgl. Abs. 2.1.2), die Wirkzusammenhänge innerhalb dieser Hierarchie als verhaltensorientierte Regelkreise zu verstehen. Die Hauptaufgabe dieser Arbeit besteht also in der Ausarbeitung von verhaltensorientierter Antizipation und Regelung sowie deren Wirkzusammenhänge im Kontext dieser Planungsebenen.

Weil ein Grundverständnis für die Planungsebenen und deren Wirkzusammenhänge zu entwickeln ist, wird im folgenden Abschnitt zunächst die Hierarchie von Verhaltensplanung und -regelung im Detail betrachtet (vgl. Abs. 2.2.1). Im Anschluss folgt das Erfassen von Ausführungsverläufen bei rollierender Verhaltensplanung, zumal dieses Vorgehen den Aufbau des Modells auf der überlagernden Ebene gewährleistet (vgl. Abs. 2.2.2). Abschließend werden die Anforderungen an die Verhaltensantizipation fokussiert und in zu lösende Teilprobleme zerlegt (vgl. Abs. 2.2.3).

[48] „*Je größer die zeitliche Reichweite des Plans ist, desto geringer ist in der Regel die Verläßlichkeit der Informationen. Daher werden bei langfristiger Betrachtung eher grobe, aggregierte Pläne und bei kurzfristiger Betrachtung genaue Detailpläne erstellt.*“ ([Sch01], S. 14, Z. 23); vgl. auch hierarchische Planung [Geb09], S. 5 ff.

[49] „*The planning horizon (e. g. one year) is divided into periods (e. g. months). At the beginning of January a plan is made that covers January to December. But only the first period, the so-called frozen period, is actually put into practice. At the beginning of the second period (February) a new plan is made considering the actual developments during the first period and updated forecasts for the future periods. The new planning horizon overlaps with the previous one, but reaches one period further [...] and so on.*“ ([FMW05], S. 83, Z. 29)

[50] Vgl. [Dan09], S. 7 f.; vgl. u. a. auch [Bak77]; [Kim98]; [Sch01], S. 32 ff.; vgl. auch rollende Planung z. B. in [Geb09], S. 18 f.

[51] Vgl. [Klö09], S. 60 f.

[52] „*Durch die Verzahnung mit der Ausführung ergibt sich der Vorteil, dass Informationen die während der Planausführung gewonnen werden, zur Auswahl der besonders relevanten alternativen Planungsverläufe genutzt werden können.*“ ([Klö09], S. 60, Z. 35)

2.2.1 Hierarchische Verhaltensplanung und -regelung

In einer hierarchischen Verhaltensplanung und -regelung würde mit jeder Ebene eine Erweiterung des Planungshorizontes und eine Vergröberung der Verhaltensplanung einhergehen.[53] Der Antizipationshorizont für das Verhalten einer überlagerten Ebene ist mit dem Planungshorizont der überlagernden Ebene gleichzusetzen. Die Verhaltensplanung der gröberen überlagernden Ebene wird zur Regelung der Verhaltensplanung der detaillierenden und überlagerten Ebene (vgl. Abb. 2.3). Dabei ist es ausreichend, sich im weiteren Verlauf dieser Arbeit auf zwei angrenzende und für die Gesamthierarchie repräsentative Ebenen zu beschränken.

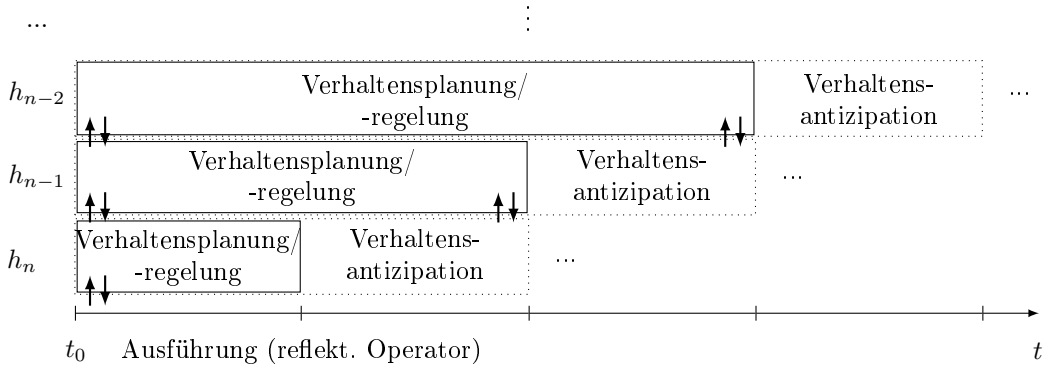


Abbildung 2.3: Hierarchie von Verhaltensantizipation und -regelung

Im nächsten Abschnitt wird zunächst ein Schema der hierarchischen Verhaltensplanung als Ansatz betrachtet (vgl. Abs. 2.2.1.1), um im darauffolgenden Abschnitt Möglichkeiten zur Integration der Verhaltensregelung in diese zu diskutieren (vgl. Abs. 2.2.1.2).

2.2.1.1 Schema einer hierarchischen Verhaltensplanung

Schneeweiß[54] unterscheidet bzgl. der hierarchischen Planung zwischen einer Top- und Basis-Ebene (vgl. Abb. 2.4). In der Top-Ebene „*wird ein zweistufiges Dynamisches Programm gelöst, das auf eine Politik führt, von der lediglich die erste Entscheidung ausgeführt wird. Die Entscheidung für die zweite Periode (Stufe) wird im Rahmen einer [rollierenden] Planung endgültig erst in der zweiten Periode berechnet und implementiert*“ ([Sch94], S. 162, Z. 34). In jedem Planungszyklus

[53] Vgl. hierarchische Planung in [Sch01], [Geb09] und [Sch94].
[54] Vgl. [Sch94].

antizipiert[55] die Top-Ebene im Vorfeld für die nächste Periode die Auswirkungen verschiedener Entscheidungsalternativen bzgl. der Basis-Ebene und bezieht diese in ihre Planung mit ein. Dazu wird ein antizipiertes Basis-Modell genutzt, das mittels aktuell zur Verfügung stehender Informationen über die Basis-Ebene konstruiert wird (vgl. Abb. 2.4). Den dafür erforderlichen Informationsfluss bezeichnet man auch als Vorkopplung[56] (vgl. Abb. 2.4, Feedforward). Das Ermitteln der optimalen Entscheidungen ist dann als ein Suchprozess auf diesem vereinfachten Modell zu verstehen, bei dem die Qualität des Ergebnisses von dem Informationsstand der Top-Ebene über die Basis-Ebene abhängt.

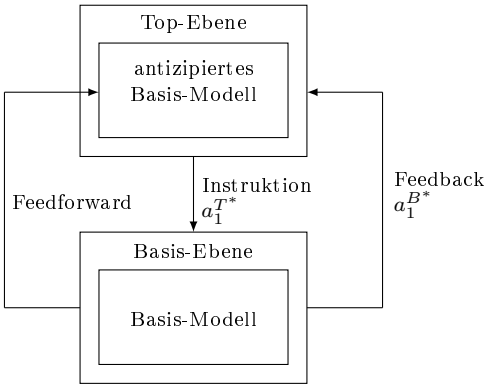


Abbildung 2.4: Allgemeines Schema hierarchischer Planung (Quelle: [Sch94])

Ist die optimale Aktion von der Top-Ebene ermittelt, so wird diese der Basis-Ebene zur Einplanung bzw. Ausführung mitgeteilt (vgl. Abb. 2.4, Instruktion $a_1^{T^*}$). Im Allgemeinen ist diese Mitteilung zunächst faktisch und löst eine Reaktion der Basis-Ebene aus (vgl. Abb. 2.4, Feedback $a_1^{B^*}$), die in eine ggf. erforderliche Aktualisierung der Planung bzw. Antizipation auf der Top-Ebene miteinzubeziehen ist (Aushandlungsprozess). Findet a priori zur Ausführung kein Aushandlungsprozess statt, ist die Instruktion unmittelbar final und die Reaktion der Basis-Ebene erfolgt ex post, indem die Auswirkungen der Ausführung an die Top-Ebene zurückfließen. Diese beiden Möglichkeiten des Rückflusses bezeichnet man auch als

[55] „*Antizipation: Hierbei handelt es sich [...] um eine Ex ante-Berücksichtigung der unteren Ebenen. Jedoch bezieht sich diese nicht auf dort vorliegende Daten, sondern auf mögliche Ergebnisse bei deren Planung. Es wird versucht, die Auswirkungen verschiedener Entscheidungsalternativen in der aktuellen Ebene auf die von daraus resultierenden Vorgaben betroffenen untergeordneten Ebenen abzuschätzen. Zu diesem Zweck wird eine stark vereinfachte (relaxierte und/oder aggregierte) Version des dort vorliegenden Entscheidungsproblems bei der Planung auf der aktuellen Ebene mitberücksichtigt.*“ ([Sch01], S. 36, Z. 8)

[56] „*Vorkopplung (feed forward): Informationen, die untere Ebenen betreffen und für die Entscheidung auf der betrachteten Ebene relevant sowie mit hinreichender Sicherheit und Genauigkeit verfügbar sind, können im Sinne eines Vorgriffs bei der Entscheidung auf der aktuellen Ebene berücksichtigt werden.*“ ([Sch01], S. 36, Z. 4)

Rückkopplung[57].

Die Abb. 2.5 zeigt die formale Darstellung der hierarchischen Planung nach Schneeweiß (ohne Aushandlungsprozesse). Das Modell der Top-Ebene M^T ist abhängig von der Feedforward-Information FF zur Basis-Ebene, einer Antizipationsfunktion $AF = AF(a^T)$ und von dem gesamten Informationsstand I_t^T der Top-Ebene zum aktuellen Zeitpunkt t. Dabei liefert die Antizipationsfunktion AF zu bestimmten Aktionen a^T mögliche Reaktionen der Basis-Ebene zurück und nimmt dabei eine zentrale Rolle in der Top-Ebene ein. Die Parameter des Modells der Basis-Ebene M^B hingegen bestehen zum einen aus der Instruktion IN der Top-Ebene und zum anderen aus dem aktuellen externen Informationsstand $I_{t'}^B$ mit $t' \geq t$. Letzteres macht deutlich, dass die Entscheidungen der Top- und Basis-Ebene nicht zur gleichen Zeit erfolgen.[58]

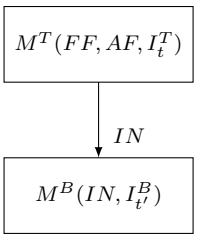


Abbildung 2.5: Formale Darstellung hierarchischer Planung (Quelle: [Sch94])

Bezieht man Aushandlungsprozesse bzw. die Rückkopplungen von Basis- zur Top-Ebene mit ein, so kann ein Regelkreis zur Verhaltensregelung in die hierarchische Planung integriert werden. Diese Einbindung der Verhaltensregelung ist Gegenstand des nächsten Abschnittes.

2.2.1.2 Integration von Verhaltensregelung

Mit der Ausführung einer eingeplanten Instruktion auf unterster Hierarchieebene[59] verändert das kognitive mechatronische System zusätzlich zu seinem eigenen Zustand die wahrgenommene Umwelt und damit auch indirekt wieder die auf der Wahrnehmung und dem Zustand basierende Verhaltensplanung der angrenzenden

[57] *„Rückkopplung (feed back): Die unteren Ebenen teilen den ihnen übergeordneten Ebenen die Ergebnisse der unter den bestehenden Vorgaben durchgeführten Planung mit. Derartige Informationen lassen sich ex post bei weiteren Planungsschritten auf der (den) oberen Ebene(n) im Hinblick auf die Verbesserung des Gesamtergebnisses einbeziehen. Somit ist es wünschenswert und häufig unumgänglich, die Planung zumindest auf den oberen Ebenen in einer rollierenden Weise mit kurzen Planungsabständen vorzunehmen, um diese Information möglichst schnell zu berücksichtigen.“* ([Sch01], S. 36, Z. 16)

[58] Vgl. [Sch94].

[59] Schneeweiß bezeichnet diese Ebene als Objekt-Ebene, in der sich das Objekt-System befindet (vgl. [Sch94], S. 163). In dieser Arbeit handelt es sich dabei um den reflektorischen Operator der OCM-Architektur des mechatronischen Systems (vgl. Abs. 2.1.1).

Basis-Ebene selbst (geschlossener Wirkungsablauf).[60] Durch die Vor- und Rückkopplungsbeziehungen der einzelnen Planungsebenen (vgl. Feedforward und Feedback in Abs. 2.2.1.1) wirkt dieser Zusammenhang in gleicher Art und Weise bis zur obersten Top-Ebene der gesamten Planungshierarchie des autonomen Systems. Damit über mehrere Planungszyklen der (rollierenden) Verhaltensplanung einer Basis-Ebene hinweg ein ökonomischer und insb. auch autonomer Betrieb gelingt, ist fortwährend das Ergebnis der Instruktionsausführung dieser Ebene in Form des tatsächlich erreichten Nutzenwertes des Zielsystems $U(x)$ (Regelgröße) von der Top-Ebene zu erfassen und im Sinne einer Angleichung an den theoretischen maximalen Nutzenwert des Zielsystems $U'(x)$ (Führungsgröße) zu beeinflussen (vgl. auch Ziele und Zielsysteme in Abs. 2.1.2).

Verhaltensregelung	(rollierende) hierarchische Verhaltensplanung
Führungsgröße	Theor. max. Nutzenwert des Zielsystems $U'(x)$
Regler	Verhaltensplanung (Top-Ebene, $M^T(FF, AF, I_t^T)$)
Stellgröße	Instruktion (Top- zur Basis-Ebene, IN)
Regelstrecke	Verhaltensplanung (Basis-Ebene, $M^B(IN, I_{t'}^B)$)
Störgröße	Externe Ziele/Umgebungseinflüsse
Regelgröße	Tats. Nutzenwert $U(x)$ des Zielsystems

Tabelle 2.1: Ansatz zur Integration von Verhaltensreglung in eine hierarchische Verhaltensplanung

Es ergibt sich der Bedarf nach einer integrierten Regelung[61], bei der die Verhaltensplanung der Basis-Ebene (Regelstrecke) in geeigneter Weise mittels Instruktionen (Stellgröße) von der Verhaltensplanung der Top-Ebene (Regler) derart gesteuert[62] wird, dass eine längerfristige Ausführung unter dem Einfluss externer Ziele und Umgebungseinflüsse (Störgröße) gemäß den übergeordneten Zielvorgaben er-

[60] „*Durch den Umweltbezug kognitiver Systeme kommt es zu Rückkopplungen mit der Umwelt. Die vom System aufgenommene Information dient zur Handlungssteuerung des Systems, und die Wahrnehmung dadurch bewirkter Veränderungen wiederum zur Kontrolle der Systemtätigkeit*“ ([GRS03], S. 20, Z. 21)

[61] „*Das Regeln, die Regelung, ist ein Vorgang, bei dem fortlaufend eine Größe, die Regelgröße (die zu regelnde Größe), erfaßt, mit einer anderen Größe, der Führungsgröße, verglichen und im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst.*“ ([DIN94], S. 7, Z. 15)

[62] „*Das Steuern, die Steuerung, ist der Vorgang in einem System, bei dem eine oder mehrere Größen als Eingangsgrößen andere Größen als Ausgangsgrößen aufgrund der dem System eigentümlichen Gesetzmäßigkeiten beeinflussen.*“ *([DIN94], S. 7, Z. 1)*

reicht wird (Verhaltensregelung). Die Verhaltensplanung der Basis-Ebene hat stets zu überprüfen, ob die Instruktionen der Top-Ebene ggf. zur Unerreichbarkeit der kurzfristigen Zielzustände führen. Die Tabelle 2.1 fasst die Integration der Verhaltensregelung in die hierarchische Verhaltensplanung zusammen.

Die Top- und Basis-Ebene sind um Komponenten zur Realisierung dieser Verhaltensregelung zu ergänzen. Die Top-Ebene hat Entscheidungen mit Auswirkungen auf zukünftiges Verhalten zu treffen, das über den begrenzten Planungshorizont der überlagerten Basis-Ebene hinausgeht (vgl. Abs. 2.1.4). Weil diese Entscheidungen die Umsetzung der übergeordneten Zielvorgaben bewirken sollen, sind Informationen über die längerfristig zu erwartende Entwicklung des Verhaltens unerlässlich.

Das erwartete Systemverhalten repräsentiert eine voraussichtliche Folge von Systemzuständen und dazwischen ausgeführten Aktionen. Genau diese gilt es auf die Erreichbarkeit der Zielvorgaben im Vorfeld zu prüfen. Im Zuge der Regelung besteht also die Notwendigkeit einer Antizipation (vgl. Abs. 2.2.1.1), die das erwartete Verhalten über den Planungshorizont der Basis-Ebene hinaus vorwegnimmt und eine Entscheidungsgrundlage liefert (Verhaltensantizipation). Diese ist dabei von einer reinen Prognose zu unterscheiden. Eine Prognose liefert Aussagen über das Eintreten von zukünftigen Ereignissen. Antizipation ist eine Abschätzung der Auswirkungen von Entscheidungsalternativen ausgehend von diesen oder weiteren in Abhängigkeit der vorweggenommenen Entscheidungsfällung zu prognostizierenden Ereignissen. Ein antizipatorisches System verwendet also Prognosen, um die Antizipation durchzuführen.[63] Es ist daher der Zugriff auf ein prädiktives Modell herzustellen, das Erfahrungswerte so abbildet, dass Prognosen und damit Verhaltensantizipation für einen längerfristigen Zeitraum realisierbar ist (vgl. antizipiertes Basis-Modell in Abs. 2.2.1.1). Die Verhaltensantizipation nimmt damit eine zentrale Rolle der Top-Ebene ein und liefert die Grundlage für die selbständige Verhaltensregelung des Systems.[64]

Zusammenfassend ist diese Verhaltensregelung mit den Grundprinzipien einer modellbasierten prädiktiven Regelung[65] (MPR) aus der Regelungstechnik für lineare bzw. nicht-lineare Systeme vergleichbar, die ebenfalls auf der Basis eines prädiktiven Modells zu einem längerfristigen Prädiktionshorizont die optimale Stellstrategie für einen kürzerfristigen Stellhorizont ermittelt. Auch dort wird in jedem Zyklus ausschließlich die Instruktion für die nächste Periode zur Ausführung fixiert (Prinzip des gleitenden Horizonts).[66]

Im nächsten Abschnitt wird im Ansatz gezeigt, wie die für das Modell der Top-

[63] Vgl. z. B. [PBCF08] S. 25 ff.

[64] „*We argue that anticipation is a key ingredient for the design of autonomous, artificial cognitive agents of the future: Only cognitive systems with anticipation mechanisms can be credible, adaptive, and successful in interaction with both the environment and other autonomous systems and humans.*“ ([PBCF08], S. 3, Z. 24)

[65] Vgl. [DP06] und [DP04].

[66] „*Man kann die Funktionsweise der MPR mit der Vorgehensweise eines Schachspieler vergleichen. Dieser spielt gedanklich verschiedene Zugfolgen durch, wobei er drei, vier oder mehr Züge im Voraus betrachtet. Den ersten der ihm optimal erscheinenden Kombination spielt er dann in der Realität.*“ ([Ada09], S. 229, Z. 10)

Ebene M^T erforderliche Feedforward-Information über die Basis-Ebene FF während der rollierenden Verhaltensplanung gewonnen werden kann.

2.2.2 Erfassen von Verhalten bei ereignisorientiert angestoßener rollierender Verhaltensplanung

Die Abbildung 2.6 skizziert das Erfassen von Ausführungsverläufen bei rollierender Verhaltensplanung. Zum Zeitpunkt t_0 führt das kognitive mechatronische System den ersten Planungsprozess mit einer gewählten bzw. von der Zielvorgabe abhängigen Planreichweite durch (vgl. Abb. 2.6, T). Unmittelbar im Anschluss beginnt die Planausführung, indem das kognitive mechatronische System fortwährend die für bestimmte eintreffende Situationen eingeplanten Aktionen des ersten Planes p_0 ausführt. Dabei entspricht die Dauer einer einzelnen Aktion einer Periode (vgl. Abb. 2.6, Δ) und der Planabstand (vgl. Abb. 2.6, D) ergibt sich aus der Anzahl der bis zur nächsten Planung ausgeführten Aktionen. Dieser Prozess wiederholt sich für die Planung von p_1 bis p_n. Der jeweils ausgeführte Teil des Planes p_i sei hier als Teilplan p_i' gegeben (vgl. Abb. 2.6).

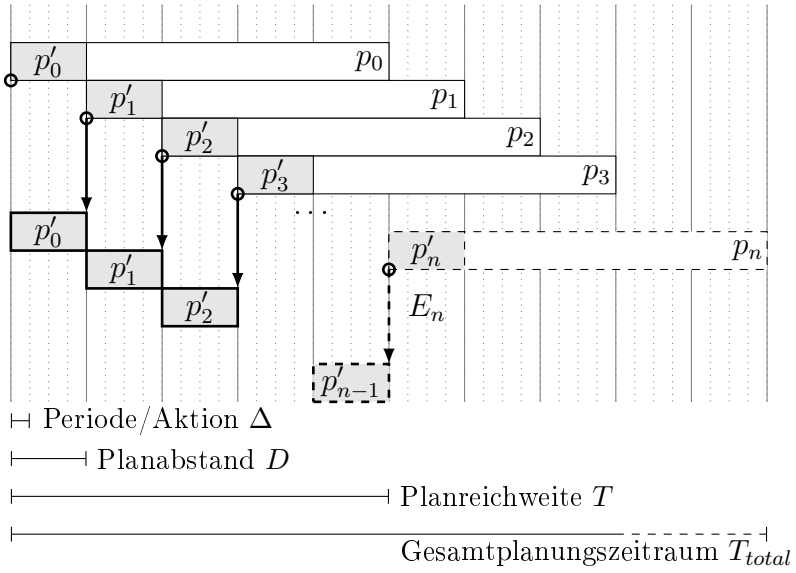


Abbildung 2.6: Planerfassung bei rollierender Verhaltensplanung

Einzelne Aktionen nehmen unterschiedlich viel Zeit in Anspruch und demzufolge variiert im Zeitverlauf der rollierenden Verhaltensplanung die Dauer einer Periode Δ_i und damit auch der Planabstand D_i (die Abb. 2.6 hingegen zeigt in vereinfachter Form Perioden und Planabstände gleicher Länge). Weiterhin ist die Übergangszeit zwischen zwei aufeinander folgenden Aktionen nicht exakt bestimmbar. Der Planabstand (bzw. Planungszyklus) ist daher nicht zeitlich festzulegen, sondern in diesem Fall durch eine maximale Anzahl m an durchzuführenden Aktionen (bzw.

Perioden) zu steuern.[67] Eine Neu- bzw. Umplanung wird nicht nach einem festgelegten Zeitintervall, sondern durch Ereignisse angestoßen (vgl. Abb. 2.6, E_n).[68] Im Falle der Verhaltensplanung (vgl. Abs. 2.1.3) sind dies im Wesentlichen zwei Ereignisse: 1. Die vorgegebene maximale Anzahl an auszuführenden Aktionen im Plan p_i ist erreicht; 2. Es existiert für den aktuellen Zustand keine eingeplante Aktion im Plan p_i. Insgesamt handelt es sich somit um eine ereignisorientiert angestoßene rollierende Planung.[69]

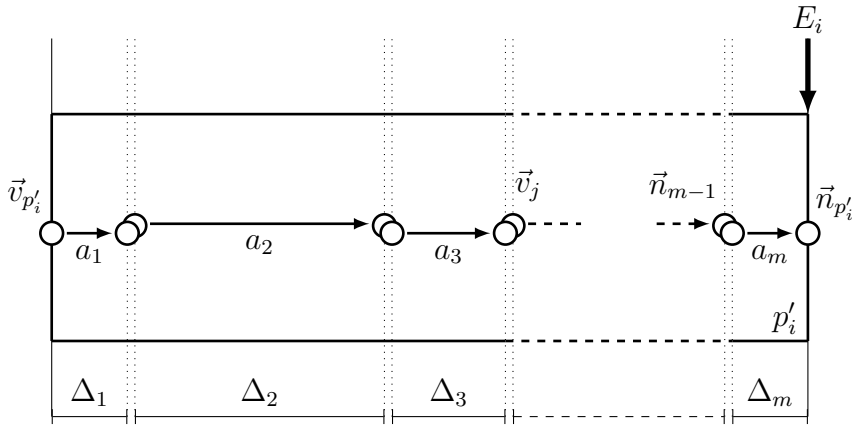


Abbildung 2.7: Ausgeführte Aktionsfolge eines Planes

Mit jedem Eintreffen eines Ereignisses E_i auf der Basis-Ebene (vgl. Abs. 2.2.1.1) kann der bereits ausgeführte Teilplan p'_i erfasst (vgl. Abb. 2.6) und damit das prädiktive Modell der Top-Ebene (vgl. Abs. 2.2.1.1) angereichert werden. Der Abbildung 2.7 ist der bis zum Eintreffen des Ereignisses E_i ausgeführte und erfasste Teil p'_i des Planes p_i mit der Aktionsfolge $a_1, \ldots, a_m$ und $1 \leq m \leq |p_i|$ zu entnehmen. Es sind die jeweiligen abgeschlossenen Perioden durch $\Delta_1, ..., \Delta_m$ abgetragen. Vor und nach jeder durchgeführten Aktion a_j sei der aktuelle Zustand des mechatronischen Systems gegeben. Der Zustand vor einer Aktion wird im Folgenden durch den Vektor $\vec{v}_j$ und der Zustand danach durch den Vektor $\vec{n}_j$ repräsentiert. Diese werden als Vor- und Nachbedingung einer Aktion bezeichnet und es gilt $\vec{v}, \vec{n} \in \mathbb{R}^n$ (vgl. Abb. 2.7). Mit dem Erfassen von ausgeführten Aktionsfolgen, die einem ex post-Feedback des Systemumfelds an die Verhaltensplanung entsprechen, werden hier die Informationsflüsse der Vor- und Rückkopplung zusammengeführt. Das Feedback dient als Feedforward-Information des prädiktiven Modells der überlagernden

[67] $|p_i|$ sei der maximal mögliche Pfad von Aktionen bzw. die Tiefe des durch die bedingten Alternativen aufgespannten Baumes im Plan p_i. Das Festlegen von $m = |p_i|$ führt zu einer Anschlussplanung indem der Plan p_i vollständig ausgeführt wird.

[68] „*A more efficient way of updating the plans is event-driven planning: A new plan is not drawn up in regular intervals but in case of an important event, e. g. unexpected sales, major changes in customer orders, breakdown of a machine etc.*“ ([FMW05], S. 83, Z. 39)

[69] „*Eine ereignisorientiert angestoßene Planung verzichtet auf einen festen Planungszyklus. Bestimmte Zustände bzw. das Eintreten bestimmter Ereignisse lösen hier eine Planung aus.*“ ([Dan09], S. 7, Z. 25)

Top-Ebene (vgl. Abs. 2.2.1.1).

Im nächsten Abschnitt werden die Anforderungen an die Verhaltensantizipation auf Basis dieses prädiktiven Modells näher betrachtet und in zu lösende Teilprobleme zerlegt.

2.2.3 Anforderungen an die Verhaltensantizipation

Nach Dubois[70] basiert das aktuelle Verhalten eines antizipatorischen Systems nicht ausschließlich auf vergangenen und/oder gegenwärtigen Ereignissen, sondern zusätzlich auf vorweggenommenen zukünftigen Ereignissen, generiert aus eben diesen vergangenen, gegenwärtigen und zukünftigen Ereignissen.[71] Weiterhin differenziert er zwischen schwacher und starker Antizipation. Schwach-antizipatorische Systeme verwenden ein prädiktives Modell von sich selbst und/oder ihres Umfelds, um Antizipation durchzuführen (exogene Antizipation). Dies entspricht im Wesentlichen der Definition nach Rosen[72], der ebenfalls die Existenz eines solchen prädiktiven Modells für antizipatorische Systeme voraussetzt. Im Gegensatz dazu zeichnen sich stark-antizipatorische Systeme durch eine striktere Form der Antizipation aus, indem sie ihre zukünftigen Zustände ausschließlich durch das Abfragen eigener interner Komponenten erzeugen (endogene Antizipation). Anders ausgedrückt: Das System kann sozusagen seine „eigene Zukunft“ selbst generieren.[73]

Davidsson[74] erklärt diesen Zusammenhang folgendermaßen: Der Folgezustand eines stark-antizipatorischen Systems könnte als eine Funktion mit vergangenen und zukünftigen Zuständen ausgedrückt werden:

$$s_{n+1} = f(s_1, s_2, ..., s_n, s_{n+1}, ..., s_k) \text{ mit } k > n.$$

Der Folgezustand eines kausalen Systems wäre hingegen nur von seinen vorherigen Zuständen bestimmt:

$$s_{n+1} = f(s_1, s_2, ..., s_n).$$

In der Regel verfügt ein kognitives mechatronisches System natürlich nicht im Vorfeld über vollständige Informationen hinsichtlich seiner zukünftigen Zustände. Ein derartiges stark-antizipatorisches System ist daher im Allgemeinen nicht umsetzbar. Was bleibt, ist die Möglichkeit, zukünftige Zuständsrepräsentationen auf Basis eines prädiktiven Modells zu prognostizieren und so ein schwach-antizipatorisches

[70] Vgl. [Dub03].

[71] Vgl. auch [Pez07] und [PBCF08].

[72] „*...; a system containing a predictive model of itself and/or of its environment, which allows it to change state an instant in accord with the model's predictions pertaining to a later instant.*“ ([Ros12], S. 313, Z. 2)

[73] Vgl. [Dub03].

[74] Vgl. [DAE94].

System zu realisieren[75]:

$$s_{n+1} = f(s_1, s_2, ..., s_n, \hat{s}_{n,1}, ..., \hat{s}_{n,k-n})$$

mit $k > n$ und $\hat{s}_{n,i}$ als prognostizierte Zustandsrepräsentation für s_{n+i}.[76]

Eine weitere Möglichkeit der Differenzierung besteht in der Betrachtung des antizipatorischen Verhaltens[77] eines Systems. Implizit-antizipatorisches Verhalten beruht nicht auf expliziten Angaben (z. B. den Vorgaben einer Prognose) über zukünftige Zustände, sondern in der direkten Übersetzung von Eingangsgrößen, ggf. in Abhängigkeit von internen Zustandsgrößen, in Aktionen des Systems (evolutionäres Verhalten).[78] Demgegenüber ist ein ergebnisorientiert-antizipatorisches Verhalten geprägt durch das Berücksichtigen von Prognosen zu Ergebnissen (z. B. im Sinne von Profit) einer jeden möglichen Aktion. Maßgeblich dabei ist, dass ausschließlich nur Prognosen zum Ergebnis in die Entscheidungsfindung über die als Nächstes auszuführenden Aktionen mit einfließen. Sensorisch-antizipatorisches Verhalten löst sich von dieser Beschränkung. Prognostizierte Sensordaten beeinflussen die aktuelle sensorische Informationsverarbeitung (und damit indirekt auch wieder die sensorische Antizipation) derart, dass z. B. erwartete gegenüber unerwarteten Sensordaten schneller verarbeitet werden können. Die letzte wohl ausgeprägteste Form ist das zustandsbasiert-antizipatorische Verhalten. In diesem beeinflusst die explizite Prognose von Repräsentationen zukünftiger Zustände die aktuellen Entscheidungen des Systems. Wie bei der sensorischen Antizipation ist ein Modell (in einfachster Weise ein explizites Prognosemodell des Systemumfelds) gegeben, das im Idealfall ebenfalls zur Durchführung einer Planung genutzt werden kann.

Das für die Verhaltensantizipation und -regelung in dieser Arbeit verwendete prädiktive Modell hat mehrere dieser Anforderungen zu erfüllen. Um die Auswirkungen von unterschiedlichen Aktionen antizipieren zu können, sollte das zugrundeliegende prädiktive Modell Informationen, zum einen über mögliche Aktionen (oder Aktionssequenzen) und zum anderen über die bei deren vorweggenommener Ausführung zu erwartenden Systemzustände in geeigneter Weise bereitstellen (**Anforderung der Verhaltensorientiertheit**). Zusätzlich muss das System in der Lage sein, die Auswirkungen der möglichen Aktionen eigenständig zu bewerten, d. h. eine

[75] „*An anticipatory system S_2 [...] which contains a model of a system S_1 with which it interacts. This model is a predictive model; its present states provide information about future states of S_1. Further, the present state of the model causes a change of state in other subsystems of S_2; these subsystems are (a) involved in the interaction of S_2 with S_1, and (b) they do not affect (i.e. are unlinked to) the model of S1. In general, we can regard the change of state in S_2 arising from the model as an adaptation, or pre-adaptation, of S_2 relative to its interaction with S_1.*“ ([Ros12] S. 317, Z. 25)

[76] Vgl. [DAE94], S. 3.

[77] Vgl. auch „*Anticipatory Behavior: A process, or behavior, that does not only depend on past and present but also on predictions, expectations, or beliefs about the future.*“ ([BSG03b], S. 4, Z. 36)

[78] „*The behavior is anticipatory in that the behavioral achitecture is predicted to be effective. For example, a genetic code is implicitly predicted (by evolution) to result in successful survival and reproduction.*“ ([BSG03a], S. 93, Z. 27)

ergebnisorientierte Antizipation realisieren zu können (**Anforderung der Ergebnisorientiertheit**).

Mit dem ständigen Erfassen von Ausführungsverläufen während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung (vgl. Abs. 2.2.2) nimmt der Informationsstand des prädiktiven Modells auf einer Top-Ebene zur überlagerten Basis-Ebene stetig zu. Weil darauf aufbauend eine fortwährende Regelung des Verhaltens der überlagerten Ebene umgesetzt werden soll, sind weitere Anforderungen zu erfüllen. Zum einen ist ein eindeutiger Bezug von erfassten Ausführungsverläufen zur aktuellen Situation herzustellen und zum anderen sollte die Antizipationsfunktion AF (vgl. Abs. 2.2.1.1) auf die wesentlichen Aspekte des Systemverhaltens fokussieren können (**Anforderung der Klassifikation**).

In den folgenden beiden Abschnitten werden diese genannten Anforderungen an die Verhaltensantizipation weiter präzisiert.

2.2.3.1 Verhaltens- und ergebnisorientierte Antizipation bei offenem Planungshorizont und mehrdimensionalen Systemzuständen

Hinsichtlich der hierarchischen Verhaltensplanung und -regelung (vgl. Abs. 2.2.1) ist es erforderlich, dass das prädiktive Modell auf der Top-Ebene eine Antizipation gewährleistet, die von der Reichweite her über den Planungshorizont der überlagerten Basis-Ebene hinausgeht (offener Planungshorizont). Im Idealfall reicht diese, bei entsprechend angereichertem Informationsstand über das Verhalten der Basis-Ebene, mindestens bis zum Planungshorizont der Top-Ebene (Antizipationshorizont, vgl. Abb. 2.8, $T_{h_{j-1}}$). Auf diese Weise lässt sich vorraussichtlich längerfristig ein Vorteil gegenüber einer reinen (nicht-hierarchischen bzw. nicht-geregelten) Verhaltensplanung (vgl. Abs. 2.1.3) auf der Basis-Ebene erzielen.

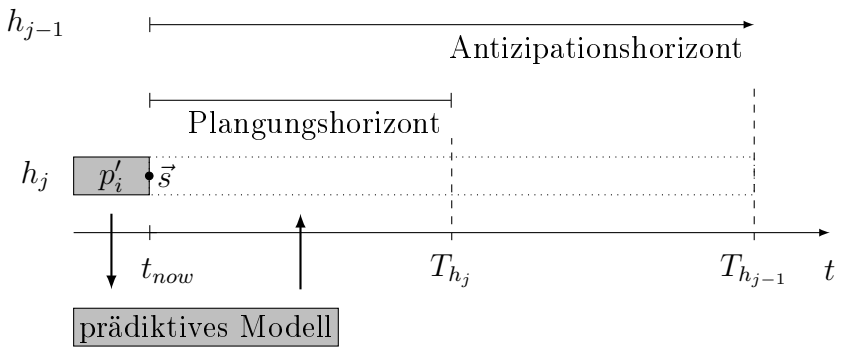


Abbildung 2.8: Verhaltensantizipation bei offenem Planungshorizont

Aus den Aktionsfolgen, die während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung erfasst werden (vgl. Abb. 2.7), ist daher ein prädiktives Modell zu konstruieren, das die Zustände (Vor- und Nachbedingungen) und Aktionen abbildet und ausgehend vom aktuellen Systemzustand $\vec{s}$ eine planungshorizontübergreifende Antizipation dieser Folgen ermöglicht (verhaltensorientierte

Antizipation). Dieser Prozess beinhaltet die Prädiktion von voraussichtlich eintreffenden Folgezuständen bei entsprechend festgelegter Aktion. Demzufolge muss das Modell experimentell bestimmte Gewichtungen abbilden können. Diese sind bei der Konstruktion des Modells aus den historischen Ausführungsverläufen zu ermitteln.

Damit neben der verhaltensorientierten auch eine zusätzliche ergebnisorientierte Antizipation gelingt, ist das Bewerten der zu erwartenden Verhaltensentwicklung unerlässlich. Beinhaltet die verhaltensorientierte Antizipation die Repräsentation von zu erwartenden Systemzuständen (vgl. Vor- und Nachbedingungen in Abs. 2.2.2), so bietet dieses die Möglichkeit zur Bewertung von gewählten Aktionen (bzw. Aktionsfolgen). Die Ausführung einer gewählten Aktion überführt eine Vorbedingung in eine Nachbedingung; genau diese Auswirkung gilt es zu quantifizieren. Weil der Systemzustand des kognitiven mechatronischen Systems in der Regel mehrdimensional ist, muss beachtet werden, dass die einzelnen Komponenten der Vor- und Nachbedingungen verschiedene Einheiten im Urbildbereich repräsentieren (z. B. Temperatur °C, Energiestand %, Standort 1 bis 10, Windgeschwindigkeit km/h usw.). Auf Grundlage dieses Ansatzes sind direkte Aufwandsabschätzungen dimensionsspezifisch im Vorfeld durchführbar (vgl. Abb. 2.9). Das Ziel besteht darin, das kognitive mechatronische System in die Lage zu versetzen, die Auswirkungen alternativer Aktionen eigenständig bewerten zu können (ergebnisorientierte Antizipation).

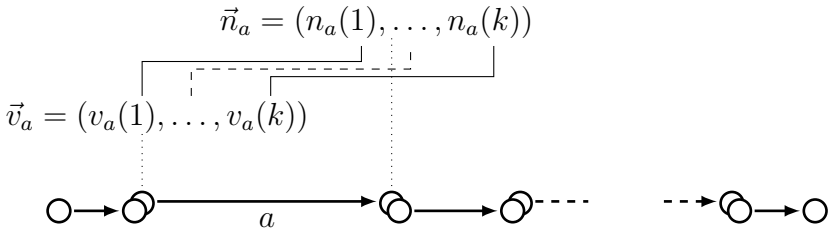


Abbildung 2.9: Dimensionen von Vor- und Nachbedingung ausgeführter Aktionen

Mithilfe dieser Bewertungsmöglichkeit und der im Rahmen des Zielsystems definierten Nutzenfunktionen (vgl. Abs. 2.1.2) kann so der Gesamtnutzen $U(x)$ zu den aktuellen Zielen für das antizipierte Verhalten bestimmt und dadurch verschiedene Aktionsfolgen im Sinne einer Präferenzordnung gegenübergestellt werden. Die Antizipationsfunktion der Top-Ebene (vgl. AF in Abs. 2.2.1.1) ist durch Lösungsverfahren auf dem prädiktiven Modell umzusetzen, die für den Antizipationshorizont eine planungshorizontübergreifende sowie optimale Strategie (Folge von Instruktionen) zur Steuerung der überlagerten Basis-Ebene ermitteln (vgl. IN in Abs. 2.2.1.1).

Insgesamt sind also bestehende prädiktive Modelle und Lösungsverfahren zur verhaltens- und ergebnisorientierten Antizipation auf die Anwendbarkeit zur Erfüllung der genannten Anforderungen zu untersuchen.

2.2.3.2 Klassifikation von Zuständen unter Berücksichtigung kontinuierlicher Prozesse

Die Sensorik des kognitiven mechatronischen Systems stellt die in Form von Signalen gegebenen Eingangsgrößen (z. B. Außentemperatur oder Krafteinwirkungen) für die Informationsverarbeitung bereit. Aufgrund der durch kontinuierliche Prozesse geprägten Umwelt können sich diese je nach Umwandlung bzgl. des Definitions- und Wertebereichs stark unterscheiden.[79] Dabei reicht die Bandbreite von zeit- und wertkontinuierlichen Signalen, analogen Signalen, bis hin zu zeit- und wertdiskreten Signalen, digitalen Signalen.[80] Bei (quasi-)kontinuierlichen Werten werden zur konkreten Zustandsbestimmung des kognitiven mechatronischen Systems, die Signalwerte z. B. durch Techniken, basierend auf Fuzzy-Mengen und Fuzzy-Logik[81], auf diskrete Zustandswerte abgebildet. Je nachdem, wie viele Werte des reellen Zahlenraumes dabei im Bildbereich abgedeckt werden, variiert die Anzahl an unterscheidbaren Vor- sowie Nachbedingungen. Während bei einer (quasi-)kontinuierlichen Abbildung – eine entsprechende Messgenauigkeit der Sensoren vorausgesetzt – die Anzahl sehr hoch ist, da sich die Werte nur infinitesimal unterscheiden (z. B. 31.324 °C, 31.325 °C, 31.336°C, ...), ist bei einer eher diskreten Abbildung die Anzahl sehr gering (z. B. Standort 1, 2, 3, ...).

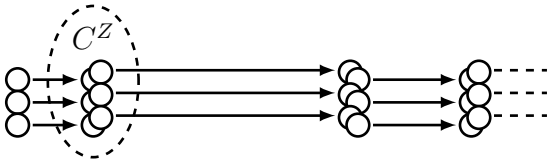


Abbildung 2.10: Klassifikation von Zuständen

Auf Grund dieser Bedinungen entstehen weitere im Zusammenhang mit der Verhaltensantizipation und -regelung in dieser Arbeit zu lösende Teilprobleme. Wird während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung einer Basis-Ebene (vgl. Abs. 2.2.1.1) fortlaufend das Verhalten erfasst (vgl. Abs. 2.2.2) und damit ein prädiktives Modell auf der überlagernden Top-Ebene (vgl. Abs. 2.2.1.1) konstruiert, so ist bei eher (quasi-)kontinuierlichen Zustandswerten die Anzahl an unterschiedlichen erfassten Zuständen sehr hoch.

Als Folge davon bleiben Repräsentationen von Zustandswiederholungen im prädiktiven Modell annähernd vollständig aus. Demnach sind die durch Beobachtung gewonnenen Gewichtungen innerhalb des Modells wenig aussagekräftig und die

[79] „*Ein Sensor wandelt eine Zustandsgröße eines technischen Prozesses, deren Qualität sich nicht als Signal eignet, in ein übertragbares, weiterverarbeitbares und registrierbares Signal um.*“ ([VDI04], S. 116, Z. 32); vgl. auch [Rod06], S. 149; [Ise08], S. 413 ff.; [Czi08], S. 61 ff.

[80] Vgl. z. B. Signalklassifizierung nach Definitions- und Wertebereich: zeit- und wertkontinuierlich (analog); zeitkontinuierlich und wertdiskret; zeitdiskret und wertkontinuierlich; zeit- und wertdiskret (digital) in [Tho06], S. 178.

[81] Vgl. z. B. Modellierung kontinuierlicher Verläufe mittels Fuzzy-Approximation in [Klö09], S. 37 ff.; vgl. auch insb. [Zad65]; [Unb08], S. 339, f.; [Rom94], S. 4 ff.; vgl. auch Fuzzy-Systeme in [Lip06], S. 245 ff.

Durchführung von Analysen zur Einschätzung zukünftiger Zustandswiederholungen nicht ertragreich. Es besteht der Bedarf, Zustände des prädiktiven Modells, die jeweils sehr ähnlich sind, in übergeordnete Datenobjekte zusammenzufassen, um mit diesen dann adäquate Analysen zu fördern. Es sind Klassen von bereits im prädikativen Modell bestehenden Zuständen zu bilden (vgl. z. B. C^Z in Abb. 2.10) und zur Betriebszeit des kognitiven mechatronischen Systems damit eine Klassifikation[82] von neu erfassten Zuständen durchzuführen. Zu diesem Zweck sind Ähnlichkeitsmaße[83] zu verwenden, die die Proximität von Zuständen quantifizieren und dabei die verschiedenen Dimensionen der Zustände mitberücksichtigen.[84] Des Weiteren ist für die Verhaltensantizipation und -regelung das Finden von sowohl situations- als auch zielkonformen Zuständen im prädiktiven Modell unter (weichen) Echtzeitbedingungen (vgl. Abs. 2.1.1) relevant. Das Ergebnis der Klassifikation kann dazu beitragen diesen Suchprozess gezielt zu unterstützen und zu vereinfachen.

Insgesamt sind also bestehende Techniken zur Klassifikation von Zuständen auf die Anwendbarkeit zur Erfüllung der genannten Anforderungen zu untersuchen.

[82] „*Klassifikation [lat.] (Klassifizierung), die systemat. Einteilung oder Einordnung von ähnl. Begriffen, Gegenständen, Erscheinungen u. a. in Klassen (Gruppen) [und Unterklassen (Untergruppen) usw.].*“ ([Lex96], S. 1813, Z. 4)

[83] Vgl. Unähnlichkeits- und Ähnlichkeitsmaße in [BPW10], S. 195 ff.

[84] „*Das Ziel der Klassenbildung besteht darin, eine Menge von Objekten in Klassen, Gruppen oder Cluster einzuteilen. Die Strukturen der Daten sollen so beschaffen sein, daß die Objekte zu einer Klasse zusammengefasst werden, die zueinander eine starke Proximität (Ähnlichkeit oder Distanz) besitzen.*“ ([Pet05], S. 26, Z. 9)

3 Stand der Technik

Im Folgenden wird ein Überblick über den aktuellen Stand der Technik mit Bezug auf die in Kapitel 2 dargelegte Problemstellung geschaffen. Es werden bestehende Arbeiten sowie Techniken betrachtet und untersucht, wie diese einen Beitrag zum Lösen der dort genannten Teilprobleme leisten können. Zu diesem Zweck werden zunächst Techniken zur weiterführenden Verhaltensplanung betrachtet (vgl. Abs. 3.1), da diese zur Umsetzung der hierarchischen Verhaltensplanung und -regelung (vgl. Abs. 2.2.1) grundlegende Ansätze liefern. Daraufhin wird die Anforderung der verhaltens- und ergebnisorientierten Antizipation (vgl. Abs. 2.2.3.1) wieder aufgegriffen und zu ihrer Erfüllung adäquate prädiktive Modelle und Lösungsverfahren untersucht (vgl. Abs. 3.2). Im Anschluss sind Techniken zur Klassifikation von Zuständen Gegenstand näherer Betrachtung (vgl. Abs. 3.3). Es soll untersucht werden, wie diese zur Erfüllung der Anforderung der Zustandsklassifikation (vgl. Abs. 2.2.3.2) beitragen können.

3.1 Techniken zur weiterführenden Verhaltensplanung

Zur Durchführung einer weiterführenden Verhaltensplanung für kognitive mechatronische Systeme in nicht-deterministischer Umgebung ist es zielführend verschiedene Techniken miteinander zu kombinieren (hybride Planungsarchitektur). Im Zusammenschluss mit einer derartigen Planungsarchitektur ist unter Verwendung von deterministischen und insb. probabilistischen Planungsmodellen (vgl. Abs. 2.1.3.1 und 2.1.3.2) bereits eine Verhaltensplanung für die in dieser Arbeit betrachteten Systeme entwickelt worden. Diese ist allerdings bisher ausschließlich für einen begrenzten Planungshorizont (vgl. hierzu auch Abs. 2.1.4) einsetzbar.[85]

Um herauszufinden, welche der bereits vorhandenen Komponenten nutzbar sind und welche zur Realisierung des Gesamtvorhabens ergänzt werden müssen, sollen im Folgenden zunächst hybride Planungsarchitekturen betrachtet werden.

3.1.1 Hybride Planungsarchitekturen

Klöpper et al.[86] beschreiben eine hybride Planungsarchitektur zur weiterführenden Verhaltensplanung für kognitive mechatronische Systeme in Form von interagieren-

[85] Vgl. [Klö09], [KSWR12], [KAA12] und [AEH+11].
[86] Vgl. [KAA12].

den Komponenten zur Planung sowie Ausführung und Überwachung von Plänen (vgl. Abb. 3.1).

Die Planung gliedert sich in die drei Bereiche: Offline-, Bedingte- und Online-Planung. Offline-Planung bezeichnet einen Planungsprozess, bei dem zunächst ein deterministischer und hinsichtlich der gegebenen Ziele bestmöglicher Plan vollständig vor der eigentlichen Ausführung erstellt wird (vgl. deterministisches Planungsmodell in Abs. 2.1.3.1). Direkt im Anschluss dient dieser Plan einer Bedingten-Planung (vgl. Abb. 3.1) als Eingabe zur weiteren Analyse. Ziel der Bedingten-Planung ist es, voraussichtliche Planabweichungen zu ermitteln und proaktiv bedingte Verzweigungen mit alternativen Planverläufen für kritische Systemzustände zu generieren.[87] Dazu verwendet diese ein zusätzliches probabilistisches Modell[88] (vgl. Abs. 2.1.3.2). Die Online-Planung (vgl. Abb. 3.1) dient als eine Art Rückfallebene, mit der unter Echtzeit eine Selektion des nächsten Operationsmodus für vorher nicht eingeplante Situationen garantiert ist.

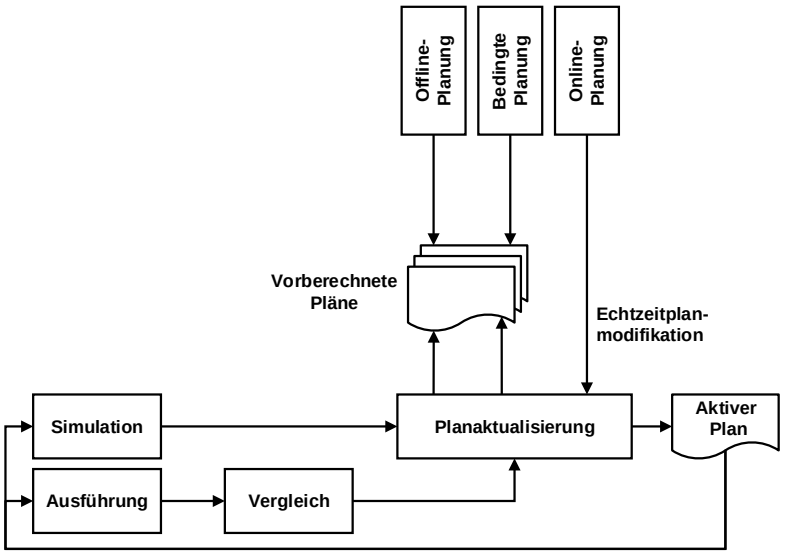


Abbildung 3.1: Hybride Planungsarchitekur (Quelle: [KAA12])

Eine Simulation des kontinuierlichen Systemverhaltens überprüft, ob die aktuelle Aktion des aktiven Plans unter den gegebenen Umfeldbedingungen ausführbar oder eine Online-Planung erforderlich ist. Mit der Ausführung des eingeplanten Operationsmodus folgt ein anschließender Vergleich von geplantem und tatsächlich erreichtem Systemzustand. Eine Planaktualisierung verarbeitet diese Informationen und prüft, ob bereits ein vorberechneter Plan verfügbar ist. Sollte dieses nicht der Fall sein, so wird mittels der Online-Planung eine Planmodifikation durchgeführt. Dies garantiert die unmittelbare Verfügbarkeit des nächsten Operationsmodus.[89]

[87] Vgl. [KSWR12].
[88] vgl. [Klö09], S. 60 ff.; vgl. insb. auch [KSWR12], S. 9 f. und [KAA12], S. 178 f.
[89] Vgl. [KAA12].

Die Planungsalgorithmen lassen sich als sogenannte *Anytime*-Algorithmen[90] implementieren. Der Planungsprozess ist daher jederzeit mit einem entsprechenden Ergebnis unterbrechbar, liefert aber mit steigender Zeit für Berechnungen eine zunehmende Qualität des Ergebnisses. Es können mehr Verzweigungen betrachtet und eine höhere Planungstiefe erreicht werden.[91]

Klöpper et al. gehen auch auf eine wissensbasierte Online-Planung (engl. *knowledge based online planning*) ein, die eine Planungsarchitektur mit zwei Bereichen bzw. Phasen umfasst, eine Offline- und Online-Phase. Während der Offline-Phase wird von einem Offline-Planer eine bestimmte Anzahl von Beispielproblemen optimal gelöst. Im Anschluss werden die erzeugten Pläne mittels eines Klassifizierers auf eine Trainingsmenge angewendet und eine entsprechende Zuordnung getroffen. In der Online-Phase kann dann das kognitive mechatronische System zu bereits bekannten Problemen auf die klassifizierten Pläne zurückgreifen. Weiterhin wird zu neuen unbekannten Problemen oder ggf. Planabweichungen nachträglich eine Aktualisierung der Wissensbasis vorgenommen.[92]

Adelt et al.[93] verwenden ebenfalls eine hybride Planungsarchitektur in ähnlicher Art und Weise. Sie zielen dabei insb. auf die Approximation von kontinuierlichen Prozessen mithilfe eines erweiterten Schemas ab. Zum einen werden zur Approximation der kontinuierlichen Effekte von Aktionen, und damit erreichbaren Folgezuständen, während der Planerstellung Simulationen zu dem Systemverhalten des mechatronischen Systems durchgeführt. Das Ergebnis besteht aus einer Anzahl von (quasi-)kontinuierlichen Werteverläufen, die auf die Erfüllbarkeit der Zielvorgaben hin überprüft werden. Zum anderen erfolgt eine Fuzzy-Approximation zum Bestimmen des Zustands der Systemumgebung sowie des Verhaltens von Teilmodulen, zu denen keine exakten physikalischen Modelle verfügbar sind (Black Box). Des Weiteren nutzen Adelt et al. auch eine Wissensbasis. Diese enthält zur Betriebszeit einplanbare Operationsmodi, die in Form von Pareto-Mengen zu schon während der Entwurfszeit des Systems gelösten kontinuierlichen Mehrzieloptimierungsproblemen für häufig eintretende Situationen bereitgestellt werden.

Auch Klöpper[94] verweist in seiner Arbeit bereits auf eine vorgelagerte Mehrzieloptimierung sowie Fuzzy-Approximation zur Integration von kontinuierlichen Prozessen.

3.1.2 Bewertung

Die Betrachtung von hybriden Planungsarchitekturen (vgl. Abs. 3.1.1) hat gezeigt, dass mithilfe eines Zusammenschlusses verschiedener Techniken bereits eine geeignete Grundlage zur Verhaltensplanung auf einer Basis-Ebene (vgl. Abs. 2.2.1.1) existiert. Auch das erforderliche Berücksichtigen der nicht-deterministischen System-

[90] Vgl. [Zil96].
[91] Vgl. [KAA12].
[92] Vgl. [KAA12], S. 180 ff.
[93] Vgl. [AEH+11].
[94] Vgl. [Klö09].

umgebung und die Integration kontinuierlicher Prozesse scheint schon weitgehend gelöst.

Zur Umsetzung einer hierarchischen Verhaltensplanung und -regelung (vgl. Abs. 2.2.1) unter langfristigen Gesichtspunkten müssen die hybriden Planungsarchitekturen um eine Top-Ebene (vgl. Abs. 2.2.1.1) in Form von zusätzlichen Komponenten erweitert werden. Weil die Top-Ebene die Verhaltensplanung (Offline-, Bedingte- und Online-Planung, vgl. Abs. 3.1.1) der Basis-Ebene gemäß der langfristig zu erwartenden Entwicklung zu beeinflussen hat, sollte diese das prädiktive Modell und eine Lösungskomponente umfassen. Unter Verwendung der Lösungskomponente kann dann auf dem prädiktiven Modell die verhaltens- und ergebnisorientierte Antizipation durchgeführt sowie die bestmögliche Instruktion (vgl. Abs. 2.2.1.1) zur Steuerung des Planungsprozesses der überlagerten Basis-Ebene bestimmt werden. Zudem ist durch das Alternieren von Planung und Ausführung das Erfassen von ausgeführten Plänen bei einer ereignisorientiert angestoßenen rollierenden Planung (vgl. Abs. 2.2.2) realisierbar. Hier ist eine entsprechende Schnittstelle zwischen der Ausführung des kognitiven mechatronischen Systems und dem prädiktiven Modell zu schaffen. Zusätzlich ist an dieser Stelle, wie anhand der Wissensbasis von Klöpper et al. in ähnlicher Art und Weise gezeigt, das Einbinden einer Klassifikation von Zuständen möglich.

3.2 Prädiktive Modelle und Lösungsverfahren zur verhaltens- und ergebnisorientierten Antizipation

Zur Integration einer Verhaltensregelung wurde die Erforderlichkeit eines prädiktiven Modells auf der Top-Ebene festgestellt (vgl. Abs. 2.2.1.1). Wie anschließend in Abschnitt 2.2.3.1 erläutert, muss dieses Modell eine Antizipation gewährleisten, die von der Reichweite her über den begrenzten Planungshorizont der überlagerten Basis-Ebene hinausgeht (offener Planungshorizont). Zudem hat dieses Zustände und Aktionen abzubilden, d. h. verhaltensorientiert zu sein, und eine Prädiktion von voraussichtlich eintreffenden Folgezuständen bei entsprechend festgelegter Aktion zu ermöglichen. Zusätzlich verlangt die Anforderung der Ergebnisorientiertheit (vgl. Abs. 2.2.3.1) das Bewerten von Auswirkungen selektierter Aktionen. Wie gefordert, ist mit einem Lösungsverfahren für das aktuelle Zielsystem, wie auch im Sinne des Gesamtnutzens $U(x)$, die bestmögliche Aktion (vgl. Instruktion IN, in Abs. 2.2.1.1) zum Steuern der Basis-Ebene zu bestimmen.

Im weiteren Verlauf dieses Abschnittes werden daher Arbeiten und Techniken zu erfolgversprechenden Modellen (vgl. Abs. 3.2.1) und Lösungsverfahren (vgl. Abs. 3.2.2) betrachtet und auf ihre Anwendbarkeit hin untersucht.

3.2.1 Modelle

Im Folgenden werden zunächst Arbeiten zu prädiktiven Modellen betrachtet und untersucht, wie diese als Basis zur Umsetzung der verhaltens- und ergebnisorientierten Antizipation dienen können.

Ghallab et al.[95] führen beispielsweise Markov-Entscheidungsprozesse[96] (engl. *markov decision process*, kurz MDP) im Bereich des automatischen Planens unter Unsicherheit ein. Dabei besteht die Grundidee darin, das Planungsproblem als Optimierungsproblem aufzufassen. Russell & Norvig[97] behandeln MDPs auf ähnliche Weise im Rahmen der künstlichen Intelligenz zur Lösung von sequentiellen Entscheidungsproblemen in stochastischer Umgebung. Sie fokussieren dabei insb. auf den Gesamtnutzen, der von der Sequenz der Entscheidungen abhängig ist. Mitchell[98] greift MDPs im Zusammenhang mit maschinellem Lernen auf. Es wird versucht auf Basis gemachter Erfahrungen das Lösen des Entscheidungsproblems zu vereinfachen. Die Anwendungsbereiche sowie Betrachtungsweisen von MDPs sind zahlreich. Einen umfassenden Einblick liefert z. B. Puterman[99].

In jedem Zeitschritt eines MDPs wird ausgehend vom aktuellen (System-)Zustand aus einer Menge von ausführbaren Aktionen gewählt. Die gewählte Aktion definiert dann Übergangswahrscheinlichkeiten zu möglichen Folgezuständen. Während das Ausführen von Aktionen in der Regel mit Kosten verbunden ist, geht im Gegensatz dazu ein Zustandswechsel mit einer Belohnung (engl. *reward*) einher. Insgesamt entsteht ein Optimierungsproblem mit dem Ziel für jeden Zustand die optimale Aktion zu finden, die den erwarteten Nutzenwert $U(x)$, bestehend aus erwarteter Belohnung abzgl. erwarteter Kosten, im Zeitverlauf maximiert (die optimale Strategie).[100] Ghallab et al. sowie Russell & Norvig beschreiben ein MDP durch folgende Elemente:

- Zustandsmenge $S = \{s_1, ..., s_n\}$,
- Aktionsmenge $A = \{a_1, ..., a_m\}$,
- bedingte Wahrscheinlichkeiten $P_a(s'|s)$, die zu jeder in einem Zustand s ausführbaren Aktion a die Übergangswahrscheinlichkeiten zu möglichen Folgezuständen s' angeben,
- Funktion $R : S \longrightarrow \mathbb{R}$, die zu jedem Zustand s eine Belohnung zurückliefert,
- und Funktion $C : S \times A \longrightarrow \mathbb{R}$, die zu jeder in einem Zustand s ausgeführten Aktion a die Kosten zurückliefert.

Die Lösung eines MDPs ist eine als Politik (oder auch Strategie, engl. *policy*) bezeichnete totale Funktion $\pi : S \longrightarrow A$, die zu jedem Zustand $s \in S$, den das System erreichen kann, die auszuführende Aktion zurückliefert. Gesucht wird die optimale Lösung bzw. Politik π^*, die den erwarteten Nutzenwert im Zeitverlauf

[95] Vgl. [GNT04], S. 380 ff.
[96] Vgl. insb. auch [Bel57a] und [How60].
[97] Vgl. [RN10], S. 645 ff.
[98] Vgl. [Mit97], S. 370 ff.
[99] Vgl. [Put05].
[100] Vgl. auch [BDH11].

maximiert.

Ein Vorteil von MDPs besteht darin, dass ausgehend vom aktuellen Systemzustand bzgl. der Reichweite einer Politik zwischen begrenztem (engl. *finite*) und unbegrenztem (engl. *infinite*) Zeithorizont differenziert werden kann. Bei fixiertem bzw. begrenztem Zeithorizont wird die Politik nur für die nächsten n Zeitperioden bestimmt, d. h. die von der Politik zu einem Zustand zurückgelieferte Aktion $\pi(s)$ kann sich im Zeitverlauf ändern (nichtstationäre Politik). Alden & Smith[101] beschreiben z. B. eine Betrachtungsweise von MDPs bei rollierendem Zeithorizont mit zeitabhängiger und für die nächste Periode gültiger Politik π_t. Im Vergleich dazu ändert sich die von einer Politik zurückgelieferte Aktion bei unbegrenztem Zeithorizont nicht mehr (stationäre Politik).

White[102] liefert eine umfassende Sammlung zu verschiedenen Arbeiten, die Anwendungsbeispiele von MDPs in der Praxis aufzeigen. Des Weiteren existieren in der Literatur verschiedene Ansätze zu Lösungsverfahren zum Bestimmen der optimalen Politik.[103] Einige der Lösungsverfahren werden im späteren Verlauf dieses Kapitels noch untersucht.

Ein partiell beobachtbarer Markov-Entscheidungsprozess[104] (engl. *partially observable markov decision process*, kurz POMDP) stellt die Verallgemeinerung eines MDPs dar.[105] Der wesentliche Unterschied zu einem normalen MDP besteht darin, dass bei einem POMDP davon ausgegangen wird, dass das System den Zustand, in dem es sich befindet, nicht exakt bestimmen kann. Stattdessen wird die Information, die über einen Zustand bekannt ist, also der beobachtbare Teil des Modells (bzw. des Systemumfelds), durch Beobachtungen (engl. *observations*) repräsentiert, die mit zustandsbedingter Wahrscheinlichkeit auftreten. Bis auf diesen Unterschied sind die Grundelemente eines POMDPs und MDPs identisch. Russell & Norvig[106] beispielsweise bezeichnen diesen Zusatz als Sensormodell. Ghallab et al. sowie Russell & Norvig beschrieben einen POMDP mit folgenden Elementen:

- Zustandsmenge S, Aktionsmenge A und bedingte Übergangswahrscheinlichkeiten $P_a(s'|s)$, Belohnungsfunktion R, Kostenfunktion C,
- Observationsmenge O
- und bedingte Wahrscheinlichkeiten $P_a(o|s) = 1$, die angeben, wie hoch die Wahrscheinlichkeit ist, dass eine Beobachtung o im Folgezustand s nach der Ausführung einer Aktion a eintritt, wobei gilt: $\sum_{o \in O} P_a(o|s) = 1$.

In diesem Modell basieren die aktuellen Beobachtungen auf der zuvor ausgeführten Aktion und dem daraus resultierenden Zustand. Der gleiche Zustand kann dabei zu unterschiedlichen Beobachtungen führen.[107] Für das System allerdings besteht ausschließlich die Möglichkeit Informationen über den eigenen Zustand aus diesen

[101] Vgl. [AS92].
[102] Vgl. [Whi93].
[103] Vgl. [RN10], S. 645 ff.; [Mit97], S. 370 ff; [GNT04], S. 380 ff.
[104] Vgl. z. B. [KLC98].
[105] Vgl. auch [RN10], S. 658 ff. und [GNT04], S. 392 ff.
[106] Vgl. [RN10].
[107] Vgl. [GNT04], S. 392.

Beobachtungen zu gewinnen.

Kaelbling et al.[108] beschreiben das Resultat als einen sogenannten *Belief*-Zustand (vom engl. belief für „Glauben"), der intern von dem System verwaltet wird und die in Form von Beobachtung gemachten Erfahrungen als Wahrscheinlichkeitsverteilung über die Zustände abbildet. Ghallab et al. definieren diese Wahrscheinlichkeitsverteilungen folgendermaßen: Seien b ein Belief-Zustand und B die Menge aller möglichen Belief-Zustände. Weiterhin sei $b(s)$ die Wahrscheinlichkeit, dass sich das System im *Belief*-Zustand b im Systemzustand s befindet, mit $0 \leq b(s) \leq 1$ und $\sum_{s \in S} b(s) = 1$. Dann führt die Ausführung einer Aktion a, ausgehend von einem *Belief*-Zustand b, zu einem neuen *Belief*-Zustand b'. Die Lösung eines POMDPs ist dann als eine Politik $\pi : B \longrightarrow A$ zu verstehen, die zu jedem Belief-Zustand b eine auszuführende Aktion a zurückliefert. Hierbei ist zu beachten, dass im Gegensatz zur Zustandsmenge S, die Menge B unbegrenzt ist, zumal diese Wahrscheinlichkeitsverteilungen mit kontinuierlichen Werten beinhaltet.[109]

Das Lösen eines POMDPs ist über den Zustandsraum der Belief-Zustände auf das Lösen eines normalen MDPs reduzierbar. Es kann gezeigt werden, dass eine optimale Politik $\pi^*(b)$ dieses MDPs einer optimalen Politik des zugehörigen POMDPs entspricht. Generell lassen sich so Verfahren zum Lösen von MDPs auch auf POMDPs anwenden. Diese sind aber aufgrund der kontinuierlichen Werte von Belief-Zuständen, und der damit einhergehenden Größe des Zustandsraumes, in der Praxis nicht anwendbar. Demzufolge existieren in der Literatur verschiedene Heuristiken zu POMDPs, die aber in der Regel keine Ausgabe einer optimalen Lösung garantieren.[110]

Butz et al.[111] beschreiben z. B. ein Rahmenwerk basierend auf POMDPs zur Erforschung verschiedener Antizipationsarten von *animats*[112]. Ein *animat* verfügt dabei u. a. über ein Zustandsmodell, das den zuvor beschriebenen *Belief*-Zuständen entspricht und einem prädiktiven Modell, das mit dem Grundmodell des POMDPs gleichzusetzen ist und zur eigentlichen Antizipation des Verhaltens dient.

Littman et al.[113] stellen 2002 eine Theorie zur prädiktiven Zustandsrepräsentation (engl. *predictive state representation*, kurz PSR) vor, die im Wesentlichen auf den Arbeiten von Jaeger[114] zu beobachtbaren Operator-Modellen (engl. *observable operator models*, kurz OMM) aufbaut. Die Grundidee einer PSR besteht darin, den Zustand eines Systems als eine Menge von Prädiktionen zu voraussichtlichen sowie beobachtbaren Ergebnissen möglicher Aktionsfolgen des Systems zu verstehen.[115] Dieser Ansatz unterscheidet eine PSR beispielsweise von einem MDP dahingehend, dass die Zustandsdynamik eines MDPs tatsächliche Zustandswechsel beschreibt

[108] Vgl. [KLC98], S. 106.
[109] Vgl. [GNT04], S. 395.
[110] Vgl. [GNT04] und [RN10].
[111] Vgl. [BSG03a].
[112] *Artificial animals*, vgl. hierzu z. B. [Wil91].
[113] Vgl. [LSS02].
[114] Vgl. [Jae97] und [Jae00].
[115] Vgl. z. B. auch [Wie07].

(historisch basierte Zustandsrepräsentation), während bei einer PSR, ähnlich wie bei einem POMDP, eine Art *Belief*-Zustand von dem System gehalten und fortwährend aktualisiert wird (rekursiv aktualisierte Zustandsrepräsentation). Dieser repräsentiert allerdings nicht direkt einen unsicheren Zustand, sondern mögliches zukünftiges Verhalten bei gegebenem vergangenen Verhalten des Systems. Das Kodieren der Systemhistorie in einem einzigen Zustand ist ggf. mit speichertechnischen Vorteilen verbunden.[116] Littman et al. zeigen u. a., wie sich eine PSR aus einem POMDP konstruieren lässt.[117]

Im Rahmen der PSR wird von einem dynamischen System ausgegangen, das Aktionen aus einer diskreten Menge A akzeptiert und Beobachtungen aus einer diskreten Menge O generiert. Zu diesem System ist die Wahrscheinlichkeitsverteilung über alle möglichen Historien (Folgen von Aktions-Beobachtungs-Paaren) gegeben durch

$$\mathbb{P}\left\{A \times O\right\}^* \longrightarrow [0,1],$$

mit $\mathbb{P}(a_1 o_1 ... a_l o_l)$ für die Wahrscheinlichkeit, dass die Beobachtungen $o_1, ..., o_l$ bei dazu alternierend ausgeführten Aktionen $o_1, ..., o_l$ genau in dieser Reihenfolge auftaucht. Die Wahrscheinlichkeit einer solchen zukünftigen Folge (auch als Test t bezeichnet) bei gegebener Historie h wird durch die bedingte Wahrscheinlichkeit

$$\mathbb{P}(t|h) = \frac{\mathbb{P}(ht)}{\mathbb{P}(h)}$$

ausgedrückt. Darauf aufbauend ist die eigentliche Repräsentation des Zustands zu einer Historie h gegeben durch einen Prädiktionsvektor

$$p(h) = [\mathbb{P}(t_1|h), ..., \mathbb{P}(t_n|h)]$$

mit Einträgen zu den bedingten Wahrscheinlichkeiten über mögliche zukünftige Tests. Littman et al. verwenden dann eine Projektionsfunktion, die zu bestimmten Tests den Prädiktionsvektor auf konkrete Wahrscheinlichkeitswerte abbildet. Sie fokussieren dabei ausschließlich lineare PSRs, d. h. mit linearen Projektionsfunktionen. Weiterhin wird gezeigt, wie der Prädiktionsvektor auf Basis neuer im Zeitverlauf auftretender Aktions-Beobachtungs-Paare aktualisiert werden kann.[118]

Singh et al.[119] veröffentlichen 2003 den ersten Algorithmus zum Lernen von PSRs. Es wird das Ziel verfolgt eine im Sinne der Statistik des dynamischen Systems hinreichend gute PSR zu bestimmen. Sie unterscheiden dabei zwischen zwei Teilproblemen: 1. Das Finden einer vorteilhaften Menge von Tests (engl. *discovery problem*); 2. Das Lernen von adäquaten Prädiktionen für diese Tests (engl. *learning problem*). Sie lassen dabei die Lösung des ersten Teilproblems offen und konzentrieren sich

[116] Vgl. [LSS02], S. 1 f.
[117] Vgl. [LSS02], S. 4 f.
[118] Vgl. [LSS02], S. 3 f.
[119] Vgl. [SLJ+03].

zunächst auf das Lernen möglichst korrekter Prädiktionen. James & Singh[120] greifen das erste Teilproblem 2004 wieder auf und veröffentlichen zusammen mit einem neuen erweiterten Lernalgorithmus eine Lösung dazu.

Im gleichen Jahr erweitern Singh et al.[121] die Betrachtungsweise der PSR von Vektoren auf eine Matrix der Systemdynamik, bei der die Zeilen möglichen Historien (Vergangenheit) und die Spalten möglichen Tests (Zukunft) entsprechen. Die Einträge dieser Matrix stellen die korrespondierenden bedingten Wahrscheinlichkeiten dar. Rudary & Singh[122] führen zudem eine neue Notation von Tests ein, sogenannte e-Tests, die es ihnen erlauben auch nicht-lineare PSR zu formulieren. Auf dieser Grundlage und zusammen mit der Formulierung einer Belohnungsfunktion gelingt es James et al.[123] schließlich sogar automatisches Planen mit PSRs umzusetzen.

3.2.2 Lösungsverfahren

Im Folgenden sind Arbeiten zu Lösungsverfahren von prädiktiven Modellen Gegenstand näherer Betrachtung. Es wird untersucht, wie damit für das aktuelle Zielsystem, und im Sinne des Gesamtnutzens $U(x)$, die optimale Aktion (vgl. Instruktion IN, in Abs. 2.2.1.1) zum Steuern der Basis-Ebene bestimmbar ist.

Das Wert-Iterationsverfahren[124] (engl. *value iteration*) ist ein iteratives Verfahren zum Lösen von MDPs (vgl. Abs. 3.2.1) und basiert auf der Annahme, dass bei optimaler Aktionswahl der Nutzenwert eines Zustands der unmittelbaren Belohnung zu diesem Zustand plus dem erwarteten diskontierten Nutzenwert des Folgezustands entspricht. Der Nutzenwert eines Zustands ist daher gegeben durch:[125]

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s').$$

Dieser Ausdruck wird nach Bellman[126] auch als Bellman-Gleichung bezeichnet. Zu jedem Zustand des MDPs existiert eine dieser Bellman-Gleichungen und die Nutzenwerte zu den Zuständen stellen eindeutige Lösungen des resultierenden Gleichungssystems dar. Das Ziel besteht darin, die Gleichungen zum Finden der Nutzenwerte möglichst simultan zu lösen. Aufgrund der Nicht-linearität der Gleichungen[127] wird dabei auf einen iterativen Ansatz zum Finden der optimalen Lösung zurückgegriffen.

Das Wert-Iterationsverfahren beginnt mit beliebigen Initialwerten für die Nutzenwerte der Zustände. In jeder Iteration i führt das Verfahren für alle Zustände das sogenannte Bellman-Update mit

[120] Vgl. [JS04].
[121] Vgl. [SJR04].
[122] Vgl. [RS04].
[123] Vgl. [JSL04].
[124] Vgl. z. B. [RN10], S. 652 ff. und [GNT04], S. 389 ff.; vgl. auch [BDH11].
[125] Vgl. [RN10], S. 652.
[126] Vgl. [Bel57b] und [Bel57a].
[127] Vgl. [RN10], S. 652.

$$U_{i+1}(s) \longleftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$$[128]

durch. Wird diese Aktualisierung oft genug ausgeführt, so konvergieren die Nutzenwerte in den Zuständen zu einer Lösung mit maximalem Gesamtnutzen. Die zugehörige Politik eines Zustands ist dann optimal und kann als $\pi^*(s)$ für einen Zustand s hergeleitet werden.[129]

Zu dem Verfahren kann gezeigt werden, dass zum Bestimmen einer optimalen Strategie eine maximale Anzahl an Iterationen erforderlich ist. In der Praxis wird jedoch das folgende Abbruchkriterium verwendet:

$$\max_{s \in S} |U_i(s) - U_{i-1}(s)| < \delta.$$

Dieses garantiert die Rückgabe einer $\delta-$optimalen Politik mit erwarteten Gesamtnutzen, der höchstens δ Einheiten vom Optimum entfernt ist.[130]

Russell & Norvig[131] gehen auch auf eine angepasste Variante des Verfahrens zum Lösen von POMDPs (vgl. Abs. 3.2.1) ein. Aufgrund der theoretisch unbegrenzten Anzahl von unterschiedlichen *Belief*-Zuständen wird dabei ein anderer Ansatz verfolgt. Im Rahmen von POMDPs ist die auszuführende Politik (bzw. Strategie) vom Grundsatz her als ein bedingter Plan zu verstehen. Kaelbling et al.[132] bezeichnen diesen auch als Strategiebaum (engl. *policy tree*). Es handelt sich dabei um eine Art Entscheidungsbaum, der einer nichtstationären Politik entspricht. Der Wurzelknoten repräsentiert die erste auszuführende Aktion, die eine Beobachtung auslöst, die wiederum die Aktion der nächsten Ebene bestimmt und so weiter. Das Wert-Iterationsverfahren prüft Ebene für Ebene, ausgehend vom aktuellen *Belief*-Zustand, mehrere dieser Strategiebäume auf den höchsten zu erwartenden Nutzenwert. In der Regel ist dafür der Raum an *Belief*-Zuständen in Regionen partitioniert, für die bestimmte Strategien gültig sind.[133] Weil POMDPs von der Komplexität her schwer zu lösen sind, gibt es in der Literatur Verfahren zur Vereinfachung. Zhang et al.[134] liefern z. B. ein aktuelles Verfahren, um die Geschwindigkeit des Wert-Iterationsverfahrens für POMDPs zu beschleunigen.

James et al.[135] zeigen bei der Umsetzung von Planen mit PSRs (vgl. Abs. 3.2.1) das Erweitern einer Variante des Wert-Iterationsverfahrens für POMDPs, die als inkrementelles Schneiden (engl. *incremental pruning*[136], kurz IP) bezeichnet wird und Serien von Nutzenfunktionen berechnet, die gegen eine optimale Lösung konvergieren. Zu diesem Zweck führen Sie zusätzlich zu Aktions-Beobachtungs-Paaren Ergebnisse (engl. *results*), vergleichbar mit Belohnungen von MDPs, für PSRs ein.

[128] Vgl. [RN10].
[129] Vgl. [RN10] und [GNT04].
[130] Vgl. [GNT04], S. 389 ff. und [RN10], S. 654 f.
[131] Vgl. [RN10], S. 660 f.
[132] Vgl. [KLC98].
[133] Vgl. [RN10] und [KLC98].
[134] Vgl. [ZLZ13].
[135] Vgl. [JSL04].
[136] Vgl. [ZL96].

Zur Herleitung greifen Sie dabei ebenfalls auf Strategiebäume zurück. Der resultierende Algorithmus heißt PSR-IP.

Das Politik-Iterationsverfahren[137] (engl. *policy iteration*) von Howard[138] ist ein alternatives Verfahren zum Finden einer optimalen Politik für MDPs. Dieses startet im Gegensatz zum Wert-Iterationsverfahren nicht mit beliebigen Nutzenwerten, sondern mit einer zufällig gewählten Politik. Ziel ist es diese iterativ zu verbessern. Das Verfahren liefert daher eine optimale Politik und keine Nutzenfunktion zurück. Dazu durchläuft dieses zwei Hauptphasen:[139]

1. Politikbewertung: Zu der aktuellen Politik π_i wird für jeden Zustand s der erwartete Nutzenwert bestimmt
2. Politikverbesserung: Für jeden Zustand s wird überprüft, ob eine bessere Aktion mit höherem erwarteten Nutzenwert existiert und die Politik entsprechend aktualisiert

Der Algorithmus terminiert, wenn die Politikverbesserung keine Veränderung bzw. Erhöhung der erwarteten Nutzenwerte mehr herbeiführt. Weil jede Iteration, bis auf die letzte, zu einer verbesserten Politik führt, und zu einer endlichen Anzahl an Zuständen eine endliche Anzahl an Aktionen existiert, ist diese Terminierung garantiert. Die Nutzenfunktion U_i der letzten Iteration stellt Lösungen zu den Bellman-Gleichungen dar und die zugehörige Politik ist damit optimal.[140] In jeder Iteration legt die aktuelle Politik π_i die durchzuführende Aktion $\pi_i(s)$ für einen Zustand s fest. Für die Politikbewertung bzw. das Bestimmen der erwarteten Nutzenwerte zu jedem Zustand ist daher eine vereinfachte und in Abhängigkeit von $\pi_i(s)$ zu lösende Bellman-Gleichung gegeben mit:

$$U_i(s) = R_i(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s').^{141}$$

Für einen kleinen Zustandsraum stellt das Politik-Iterationsverfahren einen effizienten Ansatz zum Lösen von MDPs dar. Ist der Zustandsraum allerdings größer, so wird auf eine modifizierte Variante (engl. *modified policy iteration*[142]) zurückgegriffen. Statt im Bewertungsschritt die Gleichungen exakt zu lösen, wird an dieser Stelle wiederholt eine Wert-Iteration mit einem vereinfachten Bellman-Update

$$U_i(s) \longleftarrow R_i(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$

für k Iterationen durchgeführt.[143]

Die zuvor betrachteten Verfahren aktualisieren in jeder Iteration zu jedem der Zustände die Politik bzw. den Nutzenwert gleichzeitig. Dies ist mit einer erhöhten

[137] Vgl. [RN10], S. 656 ff.; [GNT04], S. 387 f.; vgl. auch [BDH11].
[138] Vgl. [How60].
[139] Vgl. [RN10] und [GNT04].
[140] Vgl. [RN10].
[141] Vgl. [RN10], S. 657.
[142] Vgl. [PS78].
[143] Vgl. [RN10].

Komplexität der Lösungsverfahren verbunden. Littmann et al.[144] beispielsweise liefern einen konkreten Einblick in die Komplexität von Wert-Iterations- und Politik-Iterationsverfahren. In der Literatur existieren Ansätze, die eine Aktualisierung nur für eine bestimmte Teilmenge der Zustände vornehmen, nämlich für diejenigen die zu einer vorteilhaften Politik führen (asynchrone Iteration).[145]

Barto et al.[146] stellen 1995 den auf dynamischer Programmierung[147] basierenden Algorithmus RTDP (*Real Time Dynamic Programming*) vor, der auf das Finden einer bestmöglichen Nutzenfunktion unter Echtzeitbedingungen (bzw. bei einer Alternierung zwischen Planung und Ausführung) abzielt. Der Algorithmus verdankt seinen Namen der Anlehnung an die von Korf[148] definierte echtzeitorientierte heuristische Suche (engl. *real time heuristic search*), welche für jeden Zeitschritt eine Aktion wählt, die eine heuristische Funktion maximiert, dann davon abhängig den nächsten Ausgangszustand bestimmt und so sukzessive den Suchraum aufbaut (gierige Suche, engl. *greedy search*). Die Grundidee von RTDP ist es, eine derartige Vorwärtssuche ausgehend von einer Menge von Startzuständen in Richtung einer Menge von Zielzuständen durchzuführen und dabei nur für besuchte Zustände den erwarteten Nutzen zu berechnen bzw. ein asynchrones Bellman-Update vorzunehmen.

Das Finden einer optimalen Lösung sowie die Terminierung ist für dieses Verfahren im Allgemeinen nicht garantiert.[149] Barto et al. gehen zum Zeigen einer möglichen Konvergenz des Verfahrens auf einen pfadbasierten Ansatz ein (*Trial-based* RTDP).[150] Ein Pfad (engl. *trail*) repräsentiert ein Zeitintervall (bzw. eine diskrete Folge von Zuständen), in dem RTDP angewendet wird. Nach Beenden eines Pfades wird ein neuer Startzustand gewählt und das Verfahren wird für den nächsten Pfad wiederholt. Je nach Abbruchkriterium wird versucht diesen Algorithmus für möglichst viele Pfade anzuwenden, um eine Konvergenz der Nutzenwerte zu erzielen.

Bonet & Geffner[151] stellen etwas später u. a. noch eine Variante von RTDP mit der Bezeichnung RTDP-BEL (BEL für *belief*) vor. Dabei handelt es sich um einen RTDP-Algorithmus für POMDPs (vgl. Abs. 3.2.1).

3.2.3 Bewertung

In den vorangegangenen Abschnitten wurden prädiktive Modelle (vgl. Abs. 3.2.1) und Lösungsverfahren (vgl. Abs. 3.2.2) zur Realisierung einer ergebnisorientierten Antizipation untersucht.

[144] Vgl. [LDK95].
[145] Vgl. [RN10] und [BDH11].
[146] Vgl. [BBS95].
[147] Vgl. [Bel57b].
[148] Vgl. [Kor90].
[149] Vgl. [BBS95]; vgl. auch [GNT04], S. 391 f.
[150] Vgl. [BBS95], S. 103 ff.
[151] Vgl. [BH00] und [BG01].

Die hier betrachteten Modelle können grundsätzlich alle zur Abbildung und Prädiktion des Verhaltens von kognitiven mechatronischen Systemen genutzt werden, denn sie verfügen über Möglichkeiten zur Repräsentation von Zuständen und dazu wählbaren Aktionen sowie entsprechenden Wahrscheinlichkeitsverteilungen. Mit der Einführung einer Belohnungsfunktion für PSRs durch James et al.[152] ist mit allen Modellen ebenfalls eine Quantifizierung der Ergebnisse bzw. Auswirkungen von Handlungen in Form von Nutzen umsetzbar. Folglich erfüllen die Modelle die Anforderung der Verhaltens- und Ergebnisorientiertheit (vgl. Abs. 2.2.3.1).

Im Gegensatz zu MDPs verfügen POMDPs und PSRs über keine historisch basierte, sondern über eine rekursiv aktualisierte Zustandsrepräsentation mit *Belief*-Zustand bzw. prädiktivem Vektor (bzw. einer Matrix der Systemdynamik). Aufgrund der gegebenen Systemarchitektur OCM des kognitiven mechatronischen Systems (vgl. Abs. 2.1.1) und den Komponenten einer hybriden Planungsarchitektur (vgl. Abs. 3.1.1) kann davon ausgegangen werden, dass die der Planungsebene im kognitiven Operator unterlagernde Ebene bereits eine konkrete Repräsentation des aktuellen Systemzustands bereitstellt. Im einfachsten Fall handelt es sich bei dem nicht-linearen Systemverhalten also um einen voll beobachtbaren bzw. normalen MDP. Mit steigender Anzahl von Aspekten des Systemumfelds bei der Abbildung von Zuständen ist in der Realität selbstverständlich ein POMDP gegeben.

Weil POMDPs im Allgemeinen aber wesentlich komplexer sind, erscheint eine Verhaltensantizipation auf Basis von MDPs für eine erste Entwicklung als ausreichend. PSRs stellen eine interessante Alternative zu POMDPs dar. Es handelt sich hierbei jedoch um ein recht junges Forschungsgebiet und insb. Arbeiten zu echtzeitorientierten Lösungsverfahren, wie z. B. RTDP, sind noch zu leisten.

Aufgrund der Echtzeitbedingungen von kognitiven mechatronischen Systemen ist das Anwenden von RTDP zum Bestimmen der Instruktionen für die Basis-Ebene (vgl. Abs. 2.2.1.1) am ehesten zielführend.

3.3 Techniken zur Klassifikation von Zuständen

Die Anzahl von unterschiedlichen und während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung der Basis-Ebene erfassten Zustände im prädiktiven Modell der Top-Ebene kann sehr hoch werden. Um adäquate Analysen und das Herstellen von Situationsbezügen auf dem prädiktiven Modell zu gewährleisten, ergab sich als Folge die Anforderung der Klassifikation. Diese bestand zum einen aus der Klassenbildung von bereits erfassten sowie jeweils sehr ähnlichen und zum anderen aus der Klassifizierung neuer zur Betriebszeit des kognitiven mechatronischen Systems zu erfassenden Zuständen (vgl. Abs. 2.2.3.2). Im weiteren Verlauf dieses Abschnittes werden daher Arbeiten und Techniken zu erfolgversprechenden Verfahren zur Klassifikation untersucht.

Jain & Dubes[153] unterscheiden im Zuge von Klassifikationsverfahren zwischen

[152] Vgl. [JSL04].
[153] Vgl. [JD88].

verschiedenen Klassifikationstypen. Die oberste Ebene der Differenzierung bilden die nicht-exklusive (überlappende) und die exklusive Klassifikation. Eine exklusive Klassifikation liegt vor, wenn ein Datenobjekt genau einer Klasse (oder auch Gruppe, engl. *cluster*) angehörig ist. Im Gegensatz dazu kann bei der nicht-exklusiven (überlappenden) Klassifikation ein Datenobjekt mehreren Klassen zugeteilt sein.[154] Die exklusive Klassifikation unterteilt sich wiederum in die extrinsische (überwachte) und intrinsische (nicht überwachte) Klassifikation. Diese sind folgendermaßen zu unterscheiden: Eine extrinsische Klassifikation (oder auch einfach nur Klassifizierung) nutzt bereits existierende Informationen zu Clustern und zugeordneten Datenobjekten, um ggf. ein neues Datenobjekt entsprechend zu klassifizieren. Während die intrinsische Klassifikation (oder auch einfach nur Klassenbildung) die reine Bildung von Klassen auf Basis vorliegender und nicht klassifizierter Datenobjekte meint.[155] Die intrinsische und extrinsische Klassifikation lassen sich bzgl. der verwendeten Verfahren zur Durchführung einer Klassenbildung und Klassifizierung noch weiter unterteilen. Weiterführende bzw. umfassendere Gliederungen von Klassifikationsverfahren sind z. B. in den Arbeiten von Petersohn[156], Tan et al.[157], Groth[158], Krahl et al.[159] oder Han & Kamber[160] zu finden.

Einige dieser Klassenbildungsverfahren (vgl. Abs. 3.3.1) und Klassifizierungsverfahren (vgl. Abs. 3.3.2) sind Gegenstand der beiden nächsten Abschnitte.

3.3.1 Klassenbildungsverfahren

Zur Umsetzung der Klassenbildung von bereits erfassten sowie sehr ähnlichen Zuständen im prädiktiven Modell der Top-Ebene (vgl. Abb. 2.2.3.2) sollen in diesem Abschnitt gängige Klassenbildungsverfahren näher untersucht werden.

Nach Jain & Dubes[161] besteht das Ziel eines Klassenbildungsverfahrens (oder auch Clusteranalyse) im Wesentlichen darin, eine Menge von n Datenobjekten $X = \{x_1, x_2, ..., x_n\}$ zu partitionieren. Eine Partition (Clustering) ist eine Menge $P = \{C_1, C_2, ..., C_m\}$ von Teilmengen (Clustern) aus X, für die folgende Eigenschaften erfüllt sind: $C_i \cap C_j = \emptyset$ und $C_1, C_2 \cup ... \cup C_m = X$ für $i, j \leq m$ und $i \neq j$. In der Literatur werden zur Durchführung einer Clusteranalyse folgende Schritte genannt:[162]

1. Festlegen der Musterrepräsentation
2. Definition des Proximitätsmaßes
3. Durchführung des Clusterings bzw. der Gruppierung
4. Datenabstraktion (falls erforderlich)

[154] Vgl. [JMF99] und S. 281 f.; [AF07], S. 17 ff.
[155] Vgl. [JMF99] S. 265.
[156] Vgl. [Pet05].
[157] Vgl. [TSK06].
[158] Vgl. [Gro00].
[159] Vgl. [KWZ98].
[160] Vgl. [HK06].
[161] Vgl. [JD88].
[162] Vgl. [JMF99] und [JD88]; vgl. auch [AF07], [Jai10] oder [Ye03].

5. Datenauswertung (falls erforderlich)

Mit dem Festlegen der Musterrepräsentation findet eine Auswahl von Eigenschaften (engl. *features* oder *attributes*) statt, die das jeweilige Datenobjekt während der Clusteranalyse charakterisieren. Die ausgewählte Menge an Eigenschaften wird in diesem Kontext auch als Muster (engl. *pattern*) bezeichnet. Ein Muster ist ein Eigenschaftsvektor $\vec{x} = (x_1, ..., x_d)$ mit d Messwerten. Jeder Messwert x_i ist der konkrete Wert des Datenobjektes zu der Eigenschaft (oder dem Attribut), die durch Dimension i repräsentiert wird. Zu gruppierende Muster sind dann eine Menge $X = \{\vec{x}_1, \ldots, \vec{x}_n\}$ von n Eigenschaftsvektoren mit $\vec{x}_i = (x_{i,1}, \ldots, x_{i,d})$. Aus diesem Grund wird die Menge an Mustern als $n \times d$-Matrix (die sogenannte *pattern*-Matrix) angegeben. Jede Zeile dieser Matrix definiert ein Muster und jede Spalte eine Eigenschaft.[163] Die Eigenschaften können dabei unterschiedliche Typen aufweisen.[164]

Die Zuordnung der Datenobjekte zu Gruppen erfolgt durch das Anwenden eines (Un-)Ähnlichkeitsmaßes, dem sogenannten Proximitätsmaß[165]. Dieses wird in Abhängigkeit von den Datentypen der Eigenschaften des vorliegenden Musters ausgewählt. Minkowski-Metriken gelten als die bekanntesten Maße zur Bestimmung der Proximität zwischen zwei Datenobjekten und finden häufig Anwendung, wenn die Eigenschaftsvektoren kontinuierliche Messwerte aufweisen. Aus einer allgemeinen Form der Minkowski-Metrik ergeben sich als konkrete Ausprägungen verschiedene Metriken, wie z. B. die Euklidische Distanz oder City-Block-Distanz. Zu zwei gegebenen Eigenschaftsvektoren $\vec{x}_i = (x_{i1}, \ldots, x_{id})$ und $\vec{x}_j = (x_{j1}, \ldots, x_{jd})$ mit $x_{ik}, x_{jk} \in \mathbb{R}$ für $1 \leq k \leq d$ kann die Proximität durch die allgemeine Form der Minkowski-Metrik mit

$$d(\vec{x}_i, \vec{x}_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^r \right)^{\frac{1}{r}}$$

und $r \geq 0$ berechnet werden. Ist $r = 2$, so spricht man von der Euklidischen Distanz. Diese erlaubt geometrische Operationen auf den Mustern, wie z. B. Translation oder Rotation, und ist daher im Ingenieurwesen sehr beliebt. Mit $r = 1$ erhält man die City-Block-Distanz (auch Manhattan- oder Taxifahrer-Distanz genannt). In der praktischen Anwendung ist diese z. B. bei dem Clustern von Standorten zu finden. Repräsentieren alle Merkmale binäre Werte, dann wird die Manhattan-Distanz auch als Hamming-Distanz bezeichnet.[166] Ein Nachteil der Minkowski-Metriken besteht darin, dass die Proximitätswerte von Dimensionen mit hoher Skalierung ggf. die anderer Dimensionen dominieren. Eine Angleichung kann durch Normalisierung oder Gewichtung der Werte erreicht werden.[167]

Varianzen und Korrelationen zwischen Merkmalen können z. B. durch das Anwenden der Mahalanobis-Distanz[168] beseitigt bzw. standardisiert werden.[169] Für

[163] Vgl. [JMF99], [JD88] und [Pet05].
[164] Vgl. [GD92], S. 372.
[165] Vgl. [JD88], S. 11 und [BPW10], S. 195 ff.
[166] Vgl. [JD88], S. 14; [JMF99], S. 272; [AF07], S. 6; [BEPW08], S. 404.
[167] Vgl. [JMF99], S. 272.
[168] Vgl. [Mah36].
[169] Vgl. auch [AF07], S. 6 und [JD88], S. 16.

zwei gegebene Eigenschaftsvektoren ist die Mahalanobis-Distanz folgendermaßen definiert:[170]

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_i - \vec{x}_j)^T C^{-1} (\vec{x}_i - \vec{x}_j)}.$$

Dabei ist C eine Varianz-/Kovarianz-Matrix, oder allgemeiner aufgefasst, eine Gewichtung der Merkmale. Die Matrix C kann dabei unterschiedliche Spezifikationen aufweisen, von einer Diagonalmatrix mit ausschließlich Varianzen auf der Diagonalen bis hin zu einer vollständigen Varianz-/Kovarianz-Matrix mit ggf. zusätzlich zusammengelegten (sogenannten gepoolten) Varianzen und Kovarianzen.[171]

Zur Durchführung einer Clusteranalyse existieren verschiedene Clusteringtechniken. In der Literatur unterscheidet man häufig zwischen hierarchischen und partitionierenden Clusteringverfahren (vgl. Abb. 3.2). Hierarchische Clusteringverfahren erstellen eine Hierarchie geschachtelter Partitionen, während die Ausführung eines partitionierenden Clusteringverfahrens nur zu einer einzigen Partition als Ergebnis führt.[172] Zu jedem Verfahren existieren verschiedene Algorithmen, die wiederum bzgl. ihrer Umsetzung ebenfalls zu differenzieren sind. Als Beispiel lässt sich die Unterscheidung zwischen divisiven Clusteringverfahren (Top-down-Verfahren) und agglomerativen Clusteringverfahren (Bottom-up-Verfahren) nennen.[173]

Jain & Dubes[174] beschreiben hierarchisches Clustering als das Erstellen einer Sequenz von Partitionen, bei der jede Partition in der nächsten Partition der Sequenz enthalten ist. Eine Partition P ist in einer Partition Q enthalten, wenn jeder Cluster von P eine Teilmenge eines Clusters von Q ist. Q kann daher durch das Zusammenführen der Cluster von P gebildet werden. Im Initialzustand einer agglomerativen Variante stellt zunächst jedes der Datenobjekte einen einzelnen Cluster dar. Der erste Schritt umfasst das Finden der zwei Cluster mit der höchsten Proximität zueinander. Diese bilden dann gemeinsam einen neuen Cluster der nächsten Hierarchiestufe. Im zweiten Schritt folgt die Aktualisierung der Proximitätsmatrix, indem die Proximitätswerte zwischen dem neuen Cluster und den bereits bestehenden Clustern erneut berechnet werden. Die beiden Schritte werden solange wiederholt bis nur noch ein Cluster übrig bleibt. Im Gegensatz dazu würde die divisive Variante zunächst mit einem einzigen Cluster, der alle Datenobjekte enthält, beginnen und diesen immer weiter aufsplitten. Das Ergebnis des hierarchischen Clusterings kann durch ein sogenanntes Dendrogramm[175] visualisiert werden, in dem jede horizontale Ebene eine Partition repräsentiert.[176]

[170] Vgl. [AF07], S. 6; [JD88], S. 16; [BPW10], S. 339.
[171] Vgl. [BPW10], S. 339 f.
[172] Vgl. [JD88], S. 55 ff. und [JMF99], S. 274 ff.
[173] Vgl. [JMF99], S. 274; vgl. insb. auch [JD88], S. 57.
[174] Vgl. [JD88], S. 58.
[175] Vgl. [JD88], S. 59.
[176] Vgl. auch [JMF99], S. 275 ff.; [AF07], S. 9 f.

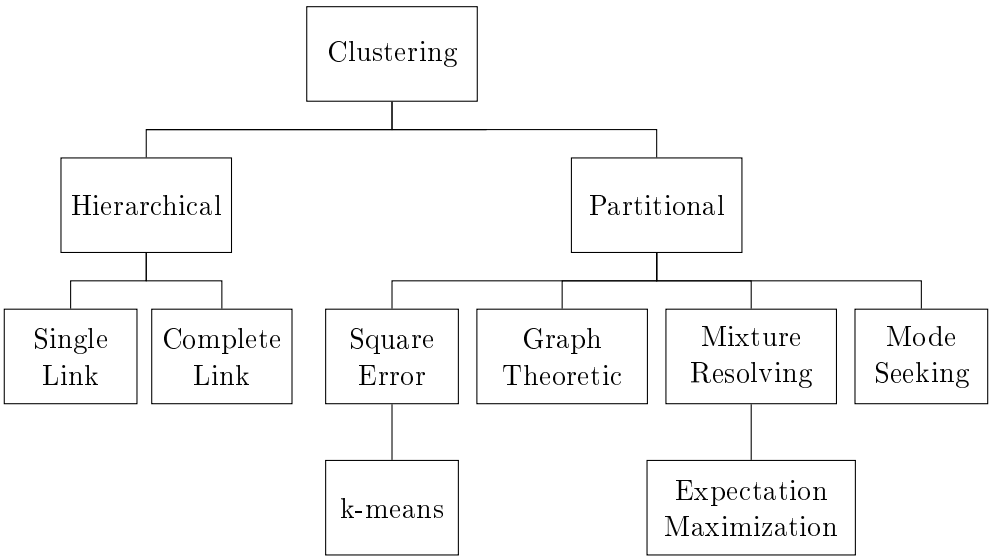


Abbildung 3.2: Eine Taxonomie von Clusteringtechniken (Quelle: [JMF99])

Zwei der bekanntesten Basistechniken sind das *Single-Linkage*-[177] und *Complete-Linkage*[178]-Verfahren (vgl. Abb. 3.2). Während bei dem *Single-Linkage*-Verfahren die Distanz zweier Cluster durch das Minimum der Distanzen aller paarweise aus diesen Clustern gewählten Merkmalsvektoren bestimmt wird (einer aus jedem Cluster), ist bei dem *Complete-Linkage*-Verfahren die Distanz abhängig von dem Maximum aller Distanzen. Jedoch werden in beiden Verfahren diejenigen zwei Cluster zu einem größeren Cluster der nächsten Hierarchieebene vereinigt, bei denen diese Distanz minimal ist. Das Ergebnis beider Verfahren unterscheidet sich dahingehend, dass die Cluster des *Complete-Linkage*-Verfahrens kompakter sind als die des *Single-Linkage*-Verfahrens.[179]

Im Gegensatz zum hierarchischen Clustering wird durch ein partitionierendes Verfahren keine Hierarchie von mehreren Partitionen, sondern nur eine einzelne Partition von Clustern erstellt. Diese Form des Clusterings wird insb. dann gewählt, wenn die vorliegende Anwendung über eine größere Menge an zu klassifizierenden Datenobjekten verfügt und dadurch der rechnerische Aufwand zum Erstellen einer hierarchischen Struktur, wie dem Dendrogramm, zu hoch ist.[180] Eines der zentralen Probleme im Zusammenhang mit partitionierenden Clusteranalysen besteht im Ermitteln der richtigen Clusteranzahl. Weil das Überprüfen sämtlicher Kombinationen zum Erzielen einer optimalen Lösung bzgl. eines gegebenen Kriteriums viel zu aufwendig ist, werden die Algorithmen in der Regel mehrfach mit verschiedenen Startzuständen ausgeführt und im Anschluss die beste Lösung als Ergebnis ausgegeben. Die Anzahl der initialen Cluster wird daher im Voraus festgelegt.[181]

[177] Vgl. [SS73].
[178] Vgl. [Kin67].
[179] Vgl. [JD88], S. 59 f.; vgl. [JMF99], S. 275 f.; vgl. [AF07], S. 9 f.
[180] Vgl. [JD88], S. 89 ff.
[181] Vgl. [Dub87] und [JD88], S. 91.

Ein einfaches und weit verbreitetes Klassenbildungsverfahren ist der *K-Means*-Algorithmus[182], der folgende Schritte durchläuft:

1. Bestimmen von K Clusterzentren (diese können z. B. aus den vorliegenden Mustern zufällig gewählt oder als neue Vektoren generiert werden)
2. Zuordnen aller Muster zu dem Clusterzentrum mit der geringsten Euklidischen Distanz
3. Neuberechnen der Clusterzentren unter Berücksichtigung der aktuellen Zuordnung
4. Ist ein Konvergenzkriterium nicht erreicht, dann gehe zu Schritt 2 (dies ist z. B. gegeben, wenn keine Neuzuordnung von Mustern oder nur eine minimale Verringerung der quadratischen Abweichung stattgefunden hat)

Ferner gibt es Verfahren zur Durchführung einer partitionierenden Clusteranalyse die auf der Graphentheorie basieren (vgl. Abb. 3.2).[183] Beispielsweise veröffentlicht Zahn[184] 1971 einen Algorithmus, der auf Grundlage eines minimalen Spannbaumes (engl. *minimal spanning tree*, kurz MST) eine Partitionierung durchführt. Dazu werden im Anschluss an die Konstruktion des Spannbaumes die längsten Kanten aus dem Graphen entfernt und so die Cluster generiert.[185] Die hierarchischen Ansätze *Single-Linkage* und das *Complete-Linkage* sind mit diesem verwandt.[186]

Ein weiterer Bereich des partitionierenden Clusterings ist durch die probabilistischen Clusteranalyseverfahren gegeben. Diese unterscheiden sich von den bereits beschriebenen deterministischen Verfahren dadurch, dass ein Datenobjekt einem Cluster mit einer bestimmten Wahrscheinlichkeit (der Zuordnungswahrscheinlichkeit) angehört. Sogenannte Mischverteilungsverfahren gehen von der Annahme aus, dass die empirische Verteilung der Datenobjekte auf einer Mischung von Wahrscheinlichkeitsverteilungen basiert. Die Aufgabe besteht darin, das Mischverhältnis und die Parameter der einzelnen Verteilungen zu schätzen und dementsprechend eine Zuordnung der Datenobjekte vorzunehmen. Diesen Prozess bezeichnet man in der Literatur als Clustering durch *Mixture-Resolving* (vgl. Abb. 3.2) oder *Mixture-Decomposition*.[187] Zum Schätzen der Parameter der zugrunde liegenden Dichtefunktion der Wahrscheinlichkeitsverteilungen wird im Allgemeinen die *Maximum-Likelihood*-Methode verwendet.[188] Dabei handelt es sich um ein allgemeines Verfahren zum Schätzen der Parameter eines statistischen Modells.[189] Ein Algorithmus zur Durchführung einer probabilistischen Clusteranalyse auf Grundlage dieser Methode ist der *Expectation-Maximization*-Algorithmus[190] (vgl. Abb. 3.2).[191]

Von diesem Verfahren klar abzugrenzen ist das Clustering durch *Mode-Seeking*

[182] Vgl. [JMF99], S. 278.
[183] Vgl. [JD88], S. 120, Z. 7.
[184] Vgl. [Zah71].
[185] Vgl. [JD88], S. 120 ff.; [JMF99], S. 279; [AF07], S. 14 ff.
[186] Vgl. z. B. [GR69] und [BH76].
[187] Vgl. [BPW10], S. 351 ff.
[188] Vgl. [JD88], S. 117 f. und [JMF99], S. 280.
[189] Vgl. [BL98].
[190] Vgl. [DLR77].
[191] Vgl. [BPW10], S. 359.

(vgl. Abb. 3.2). Währen beim *Mixture-Resolving* die Anzahl der Cluster bekannt ist, gilt es diese beim *Mode-Seeking* noch zu bestimmen. Dabei werden die Cluster als Regionen des Musterraumes mit einer hohen Dichte an Mustern aufgefasst, die durch Regionen mit geringer Dichte separiert sind. Die Identifikation von Clustern besteht in der Suche nach Regionen mit hoher Musterdichte (als *modes* bezeichnet). Jede dieser Regionen wird mit einem Clusterzentrum repräsentiert und alle *pattern* zu dem nächst gelegenen Clusterzentrum zugewiesen. In der Literatur sind viele Ansätze zum Lösen dieser Problematik zu finden.[192]

3.3.2 Klassifizierungsverfahren

Zur Umsetzung der Klassifizierung neuer zur Betriebszeit des kognitiven mechatronischen Systems im prädiktiven Modell der Top-Ebene zu erfassender Zustände (vgl. Abb. 2.2.3.2) sollen in diesem Abschnitt gängige Klassifizierungsverfahren näher untersucht werden.

Ester & Sander[193] beschreiben Klassifizierung als die Aufgabe, Datenobjekte aufgrund ihrer Attributwerte einer von mehreren vorgegebenen Klassen (Cluster) zuzuordnen. Im Gegensatz zur Klassenbildung (vgl. Abs. 3.3.1) sind hierbei die möglichen Klassen also bereits im Voraus bekannt. Es wird eine Menge von Trainingsdaten mit Attributwerten vorausgesetzt, die bereits den Klassen zugeordnet sind. Ziel ist es mittels der Trainingsdaten eine Funktion zu erlernen, die neue Datenobjekte in gleicher Art und Weise klassifiziert.[194] Klassifizierungstechniken (oder auch einfach nur Klassifikatoren) repräsentieren verschiedene Verfahren zum systematischen Aufbau eines solchen Klassifikationsmodells. Die drei bekanntesten Klassifizierungstechniken, die Bayes-Klassifikatoren, Nächste-Nachbarn-Klassifikatoren und Entscheidungsbaum-Klassifikatoren sollen im Weiteren betrachtet werden. Einen umfassenden Überblick alternativer Techniken liefern z. B. Petersohn[195], Han & Kamber[196] und Ye[197].

Bayes-Klassifikatoren[198] sind statistische Klassifikatoren, die Anwendung finden, wenn von einer nicht-deterministischen Beziehung zwischen Attributmenge und Klassenzugehörigkeit auszugehen ist. Die Klassenzuordnung eines neuen Datenobjekts ist daher, selbst bei bereits vorhandenen Trainingsdaten mit identischen Attributwerten, nicht exakt bestimmbar. Die Modellierung der probabilistischen Abhängigkeit von Attributmenge und Klassenvariable erfolgt dabei mittels bedingter Wahrscheinlichkeiten. Zur Berechnung der Wahrscheinlichkeitswerte wird u. a. auf das Bayestheorem[199] (oder auch Satz von Bayes) zurückgegriffen, das dem Klas-

[192] Vgl. [JD88], S. 118 f.
[193] Vgl. [ES00].
[194] Vgl. auch [TSK06] und [HK06].
[195] Vgl. [Pet05], S., 131 ff.
[196] Vgl. [HK06], S. 285 ff.
[197] Vgl. [Ye03], S. 13 ff.
[198] Vgl. [ES00], [Mit97], [TSK06] und [HK06].
[199] Vgl. z. B. [RN10], S. 495.

sifikator seinen Namen verdankt.

Tan et al.[200] formulieren das Grundproblem etwa wie folgt: Sei X eine Attributmenge und Y die Klassenvariable. Wenn die Klassenvariable und die Attributmenge eine nicht-deterministische Beziehung zueinander haben, dann können X und Y als Zufallsvariablen aufgefasst und ihre Abhängigkeit als bedingte Wahrscheinlichkeit ausgedrückt werden. Das Ziel besteht darin, die bedingten Wahrscheinlichkeiten $P(Y|X)$ für möglichst jede Kombination von X und Y auf Basis der Trainingsdaten zu erlernen (A-posteriori-Wahrscheinlichkeiten). Eine Klassifizierung eines neuen Datenobjekts erfolgt dann durch das Bestimmen der zugehörigen Maximalwahrscheinlichkeit.

Mitchell[201] geht von einer Menge unabhängiger Hypothesen $H = \{h_1, ..., h_l\}$ aus, die über unterschiedliche A-posteriori-Wahrscheinlichkeiten zu den Trainingsdaten verfügen, und beschreibt die Entscheidungsregel eines optimalen Bayes-Klassifikators als

$$argmax_{y_i \in Y} \sum_{h_i \in H} P(y_j|h_i)P(h_i|x).$$

Optimalität bedeutet hier, dass kein anderer Klassifikator bei gleichen Gegebenheiten im Durchschnitt eine bessere Güte in der Klassifizierung erreicht. Dieser Klassifikator maximiert die Wahrscheinlichkeit der korrekten Klassifizierung. Im Allgemeinen geht man jedoch von einem Spezialfall des optimalen Bayes-Klassifikators aus und begrenzt die Betrachtung auf genau eine gültige Hypothese. Zusammen mit Umformungen auf Grundlage des Bayestheorems erhält man dann eine vereinfachte Entscheidungsregel, die die Klasse mit der höchsten Wahrscheinlichkeit für das Auftreten des Datenobjektes bestimmt (auch als *Maximum-Likelihood*-Klassifikator bezeichnet).[202]

Das Ermitteln adäquater Werte für die bedingten Wahrscheinlichkeiten zu jeder Kombination von Klassen und Attributwerten stellt ein schwieriges Problem dar und verlangt nach einer hohen Anzahl von Trainingsdaten, auch bei einer überschaubaren Anzahl von Attributwerten. Aus diesem Grund greift man in der Praxis auf eine vereinfachte Variante, den naiven Bayes-Klassifikator, zurück. Der naive Bayes-Klassifikator geht von einem d-dimensionalen Vektor für die Attribute eines Datenobjektes aus und bestimmt die bedingten Wahrscheinlichkeiten unter der Annahme einer bedingten Unabhängigkeit zwischen Attributen und einer gegebenen Klasse. Statt für jede Kombination aus X ist es nun ausreichend die bedingte Wahrscheinlichkeit für eine Klasse zu jedem Attribut X_i einzeln zu berechnen. Dieser Ansatz reduziert die erforderliche Menge an Trainingsdaten erheblich. Der naive Bayes-Klassifikator berechnet dann die A-posteriori-Wahrscheinlichkeiten mit

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(X)}.$$

[200] Vgl. [TSK06], S. 227 ff.
[201] Vgl. [Mit97], S. 174 ff.
[202] Vgl. auch [ES00], S. 111 ff.

Weil $P(X)$ für jedes Y gleich ist, wird diejenige Klasse gewählt, die den Term $P(Y)\prod_{i=1}^{d} P(X_i|Y)$ maximiert.[203]

Hat die Annahme der bedingten Unabhängigkeit des naiven Bayes-Klassifikators bspw. aufgrund von korrelierenden Attributen keinen Bestand, so wird in der Regel eine weiterführende Modellierung der bedingten Wahrscheinlichkeiten mit Bayes'schen Netzen (engl. *Bayesian Belief Networks*[204], kurz BBN) durchgeführt, was einen wesentlich komplexeren Ansatz darstellt.[205]

Nächste-Nachbarn-Klassifikatoren[206] (engl. *nearest-neighbor classifiers*, kurz NN-Klassifikatoren) sehen von dem Bestimmen eines Klassifikationsmodells ab und verfolgen eine vereinfachte Strategie, in der der Datenbestand (bzw. die Trainingsdaten) erst mit der Klassifizierung neuer Datenobjekte betrachtet wird. Diese Art von Klassifikatoren nennt man daher auch träge Lerner (engl. *lazy learners*). Die Datenobjekte bzw. deren Attributmenge werden dabei als Punkte im d-dimensionalen Raum repräsentiert, mit d für die Anzahl der Attribute.[207]

Unter Verwendung eines Proximitätsmaßes (im einfachsten Fall die Euklidische Distanz) wird zunächst zwischen dem zu klassifizierenden Objekt $z = (\mathbf{x}', y')$ und allen Trainingsobjekten $(\mathbf{x}, y) \in D$ die Proximität berechnet. Im Anschluß folgt der Aufbau einer Liste D_z mit den k-nächsten Nachbarn (engl. *k-nearest neighbors*, kurz kNN), die den k Datenobjekten aus der Trainingsmenge D mit der geringsten Proximität zum zu klassifizierenden Datenobjekt entsprechen. Abschließend findet die Klassenzuordnung anhand der überwiegenden und bereits gegebenen Klassenzuordnung der k-nächsten Nachbarn statt.[208]

Unterschiedliche Wertebereiche der Attribute sowie die Anzahl k der einzubeziehenden nächsten Nachbarn nehmen signifikant Einfluss auf die Klassifizierungsgüte des Verfahrens. Ist die Anzahl k zu gering gewählt, so führt dies zum erhöhten Einfluss von Ausreißern. Ein zu hohes k hingegen bewirkt, dass ggf. zu viele Objekte anderer Klassen mit einbezogenen werden.[209] Die richtige Anzahl kann z. B. experimentell gewonnen werden, indem die Fehlerrate der Klassifizierung für verschiedene k miteinander verglichen wird.[210] Tan et al.[211] und Mitchell[212] schlagen bspw. die Gewichtung eines jeden nächsten Nachbarn in Abhängigkeit zu seiner Proximität zum zu klassifizierenden Datenobjekt vor. Damit soll bewirkt werden, dass weiter entfernte Datenobjekte weniger Einfluss auf die Klassenzuordnung haben als nähere. Han & Kemper[213] verweisen auf eine Min-Max-Normalisierung zur Transformation der Attributwerte auf einen Wertebereich $[0, 1]$. Diese soll verhindern, dass

[203] Vgl. [TSK06].
[204] Vgl. z. B. [Mit97], S. 184 ff.
[205] Vgl. [TSK06], S. 240 ff. und [HK06], S. 315 ff.; vgl. auch [Gro00], S. 80 ff.
[206] Vgl. [Mit97], S. 231 ff.; [ES00], S. 119 ff.; [TSK06], S. 223 ff.; [HK06], S. 348 ff.
[207] Vgl. [TSK06].
[208] Vgl. [TSK06], S. 225 und [Mit97], S. 232.
[209] Vgl. [ES00], S. 122.
[210] Vgl. [HK06], S. 349.
[211] Vgl. [TSK06].
[212] Vgl. [Mit97].
[213] Vgl. [HK06].

Attribute mit weiträumigerem die mit schmalerem Wertebereich übergewichten.

Das Anwenden von NN-Klassifikatoren ist im Allgemeinen mit hoher Laufzeit verbunden. Bei einer Trainingsmenge von $|D|$ Datensätzen und $k = 1$ sind zur Klassifizierung eines neuen Datenobjektes $|D|$ Vergleiche erforderlich. Durch Vorsortieren und Arrangieren der Trainingsmenge in Suchbäumen, z. B. mit R^*-Bäumen für niedrige Dimensionen und X-Bäumen für höhere Dimensionen, kann die Laufzeit auf $O(log(|D|))$ reduziert werden.[214]

Im Gegensatz zu den Nächste-Nachbarn-Klassifikatoren wird bei der Klassifizierung von Datenobjekten mit Entscheidungsbaum-Klassifikatoren[215] im Vorfeld eine Struktur in Form eines Entscheidungsbaumes (engl. *decision tree*) aufgebaut. In einem Entscheidungsbaum repräsentiert jeder innere Knoten einen Test auf einem Attribut, jede Verzweigung das Ergebnis eines Tests und jeder Blattknoten eine Klasse (auch als Zielgröße bezeichnet). Während der Klassifizierung eines neuen Datenobjekts durchläuft dieses den Entscheidungsbaum, ausgehend von der Wurzel, auf einem durch die Ergebnisse der Tests bestimmten Pfad bis hin zu einem Blattknoten, der schließlich die Klassenzuordnung festlegt. Entscheidungsbäume lassen sich dabei in Klassifikations- und Regressionsbäume unterscheiden. Während bei qualitativen Zielgrößen bzw. Klassenvariablen der Entscheidungsbaum als Klassfikationsbaum bezeichnet wird, so spricht man im quantitativen Fall von einem Regressionsbaum. Dabei ist die Struktur der Bäume grundsätzlich identisch.[216] Die Hauptaufgabe besteht nun darin, aus den Trainingsdaten einen adäquaten Entscheidungsbaum zu konstruieren, der zu einer bestmöglichen Klassifizierung zukünftiger Datenobjekte führt.

Pertersohn[217] bspw., liefert einen umfassenden Überblick zu Entscheidungsbaumverfahren zur Kontruktion von sowohl Klassifikationsbäumen als auch Regressionsbäumen. Bekannte Verfahren sind z. B. ID3 (*Iterative Dichomiser*) von Quinlan[218], der Nachfolger C4.5[219] oder CART[220] (*Classification and Regression Trees*). Der Basisalgorithmus ID3 stellt einen Top-Down-Ansatz dar, bei dem damit begonnen wird für die Wurzel das im Sinne der Klassifikationsgüte beste Attribut zu bestimmen. Zu diesem Zweck wird mit statistischen Tests ermittelt, wie gut jedes Attribut alleine die Trainingsdaten klassifizieren kann. Gängige Verfahren sind z. B. Informationsgewinnung (von ID3 verwendet) oder der Gini-Index (von CART verwendet). Das ermittelte Attribut definiert dann den ersten Test an der Wurzel. Im Anschluss findet für mögliche Werte (bzw. im (quasi)-kontinuierlichen Fall Wertebereiche) des Attributs das Erzeugen von Nachfolgeknoten statt. Sodann werden die Trainingsdaten entsprechend ihrer Attributwerte den Nachfolgeknoten mit den korrespondierenden Werten (oder Wertebereichen) zugeordnet (engl. *split*). Je nachdem, ob ein

[214] Vgl. [HK06] und [ES00].
[215] Vgl. [ES00], S. 126 ff.; [BV08], S. 273 ff.; [Mit97], S. 52 ff.; [HK06], S. 291 ff.
[216] Vgl. [Alp08] und [BV08].
[217] Vgl. [Pet05], S. 136 ff.
[218] Vgl. [Qui86].
[219] Vgl. [Qui93].
[220] Vgl. [BFOS83].

diskreter oder (quasi-)kontinuierlicher Wertebereich vorliegt oder die Anforderung eines Binärbaums besteht, werden hierbei verschiedene Regeln angewendet (engl. *splitting rules*). Der gesamte Prozess wird dann an jedem Nachfolgeknoten mit den dort zugeordneten Trainingsdaten wiederholt, bis eine vollständige Klassifizierung vorliegt oder alle Attribute verwendet wurden.[221]

Ein zentrales Problem bei der Klassifizierung mit Entscheidungsbäumen ist Überspezialisierung (engl. *overfitting*). Sind die Trainingsdaten fehlerhaft, z. B. aufgrund von Rauschen oder Ausreißern, oder stellen diese keine repräsentative Stichprobe dar, so kann das Anwenden von Verzweigungsregeln zu einem Entscheidungsbaum führen, der von der tatsächlichen optimalen Klassifizierung der Gesamtmenge abweicht. D. h., es existiert ein alternativer Entscheidungsbaum, der zwar für die Trainingsmenge eine geringere, aber für die Gesamtmenge eine höhere Klassifikationsgüte als der überspezialisierte Entscheidungsbaum aufweist.[222] Um Überspezialisierung zu vermeiden, werden z. B. Techniken zum Schneiden (engl. *pruning*) von Entscheidungsbäumen verwendet, die entweder im Vorfeld verhindern, dass bestimmte Verzweigungsstellen in dem Baum gebildet werden (*prepruning*), oder nachträglich den Entscheidungsbaum durch Reduktion von Teilbäumen auf einen Blattknoten ausdünnen (*postpruning*). Grundsätzlich spielt die Größe des Entscheidungsbaumes bei dieser Art von Klassifikatoren eine wichtige Rolle. Sie entscheidet darüber, ob ein Entscheidungsbaum vollständig im Speicher gehalten oder durch aufwendige Operationen Teile des Baumes nachgeladen werden müssen. Algorithmen, die Techniken zur Vorsortierung verwenden, oder den Entscheidungsbaum auf alternative Datenstrukturen abbilden, sind Möglichkeiten zur Reduzierung dieses Problems.[223]

3.3.3 Bewertung

In den vorangegangenen Abschnitten wurden potentielle Verfahren zur Klassifikation von Zuständen untersucht. Dabei wurde zwischen Klassenbildungsverfahren (vgl. Abs. 3.3.1) und Klassifizierungsverfahren (vgl. Abs. 3.3.2) unterschieden.

Mit einer quantitativen Musterrepräsentation von Zuständen des kognitiven mechatronischen Systems, zum Abbilden der (quasi-)kontinuierlichen Zustandswerte auf Attributwerte, kann auf dem aktuellen Datenbestand eine Clusteranalyse durchgeführt und das Bilden noch unbekannter Klassen zu ähnlichen Systemzuständen erreicht werden. Der Datenbestand besteht dabei aus den erfassten Plänen der Basis-Ebene (vgl. Abs. 2.2.2). Weil die Zustandsattribute des kognitiven mechatronischen Systems über Dimensionen mit unterschiedlichem Wertebereich verfügen (und ggf. sogar korrelieren), ist das Verwenden einer Minkowski-Metrik und insb. der euklidischen Distanz als Proximitätsmaß nicht ausreichend. Zum Erzielen adäquater Klassifikationsergebnisse ist eine Gewichtung, oder eine Normalisierung von

[221] Vgl. [Mit97] und [HK06].
[222] Vgl. [Mit97] und [HK06].
[223] Vgl. [HK06], S. 304, ff.

Varianzen und Korrelationen wie bei der Mahalanobis-Distanz, unabdingbar. Darüber hinaus ist so steuerbar, welche Zustandsattribute in die Analyse mit einfließen. Als Clusteringtechnik empfiehlt sich ein partitionierendes Verfahren, wie z. B. der *K-Means*-Algorithmus, zumal sich das Ergebnis eines hierarchischen Verfahrens in Form eines Dendrogramms aufgrund der ggf. hohen Datenmenge an erfassten Zuständen nur schwer darstellen lässt.

Sind die Zustandsklassen im Vorfeld gebildet (oder alternativ festgelegt) worden, dann ist eine analoge Klassifizierung von Zuständen während der Betriebszeit des kognitiven mechatronischen Systems umsetzbar. Bayes-Klassifikatoren hätten in diesem Fall den Nachteil, dass die bedingten A-postiori-Wahrscheinlichkeiten fortlaufend bestimmt bzw. aktualisiert werden müssten. Zudem würde eine mögliche Korrelation von Attributen und der damit einhergehenden Formulierung von Bayes'schen-Netzen diesen Prozess noch erheblich erschweren. Eine Klassifizierung mittels Entscheidungsbaum-Klassifikatoren umfasst gleich mehrere Teilprobleme, wie z. B. das Bestimmen der erlaubten Größe des Entscheidungsbaumes, das Behandeln von (quasi)-kontinuierlichen Attributwerten, das Auswählen geeigneter Attribute für Tests bzw. Verzweigung an inneren Baumknoten oder das Problem der Überspezialisierung. Zwar lösen weiterführende Verfahren, wie z. B. C4.5, viele der genannten Teilprobleme, dennoch scheint eine Anpassung für die Klassifizierung der Zustände von kognitiven mechatronischen Systemen unter Echtzeitbedingungen zu aufwendig. Daher stellt hier den zielführendsten Ansatz eine Klassifizierung auf Basis eines k-Nächste-Nachbar-Klassifikators dar. Zum einen kann dieses Verfahren unmittelbar mit dem gleichen Proximitätsmaß, das auch für das partitionierenden Clusteringverfahren verwendet wird, realisiert werden. Dies schafft ein einheitliches Zusammenspiel von Klassenbildung und Klassifizierung. Zum anderen kann so auf ein konkretes Klassifikationsmodell verzichtet werden. Der Nachteil bzgl. der hohen Komplexität des Verfahrens lässt sich mit Indexstrukturen wie z. B. R-Baum reduzieren. Diese können gleichzeitig zur effizienten Suche von Ausgangszuständen für die Verhaltensantizipation genutzt werden.

4 Zu leistende Arbeit

Das Ziel dieser Arbeit besteht in der Realisierung einer Verhaltensantizipation und -regelung kognitiver mechatronischer Systeme bei langfristiger Planung und Ausführung. In der Problemstellung (vgl. Kap. 2) wurde als Ansatz zur Umsetzung eine hierarchische Verhaltensplanung und -regelung mit einer Erfassung von ausgeführten Plänen bei ereignisorientiert angestoßener Verhaltensplanung vorgeschlagen (vgl. Abs. 2.2.1). Zudem fand die Spezifikation der von dieser Lösung zu erfüllenden Anforderungen mit Bezug auf die erforderliche Verhaltensantizipation statt (vgl. Abs. 2.2.3). Da mit hybriden Planungsarchitekturen (vgl. Abs. 3.1.1) bereits eine Verhaltensplanung für die Basis-Ebene (vgl. Abs. 2.2.1.1) der hierarchischen Verhaltensplanung und -regelung existiert (vgl. Abs. 3.1.2), kann die zu leistende Arbeit im weiteren Verlauf auf die Verwirklichung der Top-Ebene (vgl. Abs. 2.2.1.1) sowie einer entsprechenden Schnittstelle konzentriert werden.

Für das prädiktive Modell auf der Top-Ebene ergibt sich die Anforderung der Verhaltens- sowie Ergebnisorientiertheit (vgl. Abs. 2.2.3.1), diese soll zum einen das Abbilden von Zuständen sowie Aktionen und zum anderen das Bewerten von Auswirkungen selektierter Aktionen während der Verhaltensantizipation ermöglichen. Wie im Stand der Technik bewertet, erfüllen MDPs diese Anforderung und stellen damit eine echte Alternative zum Aufbau eines prädiktiven Modells auf der Top-Ebene dar (vgl. Abs. 3.2.3). Weil während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung der Basis-Ebene fortlaufend ausgeführte Pläne im prädiktiven Modell der überlagernden Top-Ebene zu erfassen sind (vgl. Abs. 2.2.2), müssen diese in Echtzeit zum MDP hinzugefügt werden können. Zum Steuern des Umfangs der betrachteten Historie und Realisieren eines mitlaufenden Zeitfensters ist dazu analog das Entfernen von alten Plänen aus dem MDP zu ermöglichen. Da vor dem Anwenden beider Methoden zu überprüfen ist, ob bereits identische Planrepräsentationen im MDP vorhanden sind und damit lediglich eine Aktualisierung von Gewichtungen[224], oder andernfalls eine Erweiterung bzw. Reduzierung der Zustände und Aktionen des MDPs erforderlich ist, muss zusätzlich eine Suche nach Plänen im MDP umgesetzt werden. Das erste Arbeitspaket (**AP1**) besteht daher in der Konzeption eines adaptiven MDPs mit Methoden zum Suchen, Hinzufügen und Entfernen von Plänen (Schnittstelle Basis- zur Top-Ebene).

Das Berücksichtigen von (quasi-)kontinuierlichen Zustandswerten des kognitiven mechatronischen Systems und die daraus resultierende hohe Anzahl von unterschiedlichen Zuständen im prädiktiven Modell der Top-Ebene führen zur Anfor-

[224]Im weiteren Verlauf dieser Arbeit werden auf diese Weise, d. h. durch Beobachtung gewonnene Werte, nicht mehr als Wahrscheinlichkeiten, sondern als Gewichte bezeichnet. Dies soll die experimentelle von der theoretischen Wahrscheinlichkeit abgrenzen.

derung der Klassifikation (vgl. Abs. 2.2.3.2). Diese soll sowohl adäquate Analysen sowie das Herstellen von Situationsbezügen mittels Klassenbildung als auch eine Klassifizierung neuer zur Betriebszeit zu erfassender Pläne sicherstellen. Zu diesem Zweck ist der adaptive MDP um Zustandsklassen – jeder Zustand im MDP repräsentiert nun eine Klasse von vielen ähnlichen Zuständen – zu erweitern. Die Bewertung der Klassenbildungsverfahren in Abschnitt 3.3.3 hat ergeben, dass es sich bei dem partitionierenden *k-Means*-Algorithmus um ein geeignetes Verfahren zum Erreichen der geforderten Klassenbildung handelt. Zur Durchführung der Klassenbildung ist zuvor ein Proximitätsmaß zu entwickeln, das die Ähnlichkeit der Zustände unter Einbezug der unterschiedlichen Dimensionen quantifiziert. Zum Steuern des Analysespektrums ist hierbei die Möglichkeit einer Gewichtung einzelner Dimensionen unabdingbar. Das zweite Arbeitspaket (**AP2**) besteht daher aus der Konzeption einer Erweiterung des adaptiven MDPs um Zustandsklassen, der Entwicklung eines spezifischen Proximitätsmaßes und dem Umsetzen einer an *k-Means* angelehnten Clusteranalyse.

Zur Klassifizierung während der Betriebszeit wurde im Stand der Technik der *k-Nearest-Neighbor*-Algorithmus als zweckmäßig bewertet (vgl. Abs. 3.3.3). Damit auf den gebildeten Zustandsklassen des erweiterten MDPs diese Klassifizierung gelingt, ist eine klassifikationsbasierte Erweiterung der korrespondierenden Methoden Suchen (Klassenermittlung), Hinzufügen (Klassifizierung) und Entfernen (Deklassifizierung) von Plänen zu leisten. Das dritte Arbeitspaket (**AP3**) umfasst daher die Konzeption eines an *k-nearest-neighbors* angelehnten MDP-Klassifikators mit der bereits genannten Anpassung der Methoden.

Wie in der Problemstellung gefordert, ist zur Komplettierung des Regelkreises, für das aktuelle Zielsystem (vgl. Abs. 2.1.2), und im Sinne des Gesamtnutzens U(x), die aus Sicht der Top-Ebene langfristig optimale Instruktion (vgl. IN, in Abs. 2.2.1.1) zum Steuern der überlagerten Basis-Ebene zu bestimmen (Schnittstelle Top- zur Basis-Ebene). Im Stand der Technik wurden Lösungsverfahren für MDPs untersucht. Das Echtzeit-Iterationsverfahren RTDP stellte einen vielversprechenden Ansatz zum Bestimmen der maximalen Nutzenwerte und damit optimalen Aktionen in Zuständen des MDPs unter den Echtzeitbedingungen des kognitiven mechatronischen Systems dar (vgl. Abs. 3.2.3). Dabei ist zu berücksichtigen, dass RTDP nur unter bestimmten Bedingungen eine optimale Lösung bzw. die Terminierung garantiert. Zum einen kann ein simulativer Vergleich mit garantiert optimalen Lösungsverfahren, wie bspw. dem Werte-Iterationsverfahren, eine Aussage über die Qualität von RTDP in konkreten Anwendungsfällen liefern und zum anderen durch die Begrenzung der Reichweite von Suchpfaden (*trails*) die Terminierung sichergestellt werden.

Weil in dieser Arbeit für den langfristig ökonomischen Betrieb die Minimierung des Ressourcenverbrauchs in Abhängigkeit vom aktuellen Zielsystem im Vordergrund steht, und die Aktionen sowie Zustände des MDPs nun erfassten Plänen bzw. Zustandsklassen entsprechen, ist im vierten Arbeitspaket (**AP4**) vorbereitend eine Kostenbewertung zum Bestimmen der direkten Kosten von Plänen und der zu erwartenden direkten Kosten in Zustandsklassen zu konzipieren.

Arbeitspaket	Zu leistende Arbeit
AP1	Konzeption eines adaptiven MDPs mit Methoden zum Suchen, Hinzufügen und Entfernen von Plänen
AP2	Konzeption einer Erweiterung des adaptiven MDPs um Zustandsklassen, eines spezialisierten Proximitätsmaßes und einer an *k-means* angelehnten Clusteranalyse
AP3	Konzeption eines an *k-nearest-neighbors* angelehnten MDP-Klassifikators mit angepassten Methoden zum Suchen (Klassenermittlung), Hinzufügen (Klassifizierung) und Entfernen (Deklassifizierung) von Plänen
AP4	Konzeption einer Kostenbewertung zum Bestimmen der direkten Kosten von Plänen und der zu erwartenden direkten Kosten in Zustandsklassen
AP5	Konzeption einer verhaltens- und ergebnisorientierten Echtzeit-Antizipation auf Basis von RTDP mit Methoden zum Bestimmen von situations- und zielkonformen Zustandsklassen, zur randomisierten Selektion von Folgezustandsklassen und zum Ermitteln von kostenminimalen Instruktionen
AP6	Konzeption einer Regelung von Verhaltensplanung durch Instruktionen und einer Integration in die hierarchische Verhaltensplanung
AP7	Validierung der Methoden und Evaluation der Ergebnisse

Tabelle 4.1: Zusammenfassung der zu leistenden Arbeit

Mit dem Durchführen der Antizipation sind zum aktuellen System- und Zielzustand des kognitiven mechatronischen Systems situations- und zielkonforme Zustandsklassen im erweiterten adaptiven MDP zu bestimmen, zumal RTDP eine Vorwärtssuche ausgehend von einer Menge von Startzuständen in Richtung einer Menge von Zielzuständen durchführt. Zudem wählt RTDP in jeder Iteration zufällig den Folgezustand in Abhängigkeit von den durch die kostenminimale Aktion festgelegten Gewichtungswerte. Aus diesem Grund findet im fünften Arbeitspaket (**AP5**) die Konzeption einer verhaltens- und ergebnisorientierten Echtzeit-Antizipation auf Basis von RTDP mit Methoden zum Bestimmen von situations- und zielkonformen Zustandsklassen zur randomisierten Selektion von Folgezustandsklassen und zum Ermitteln von kostenminimalen Instruktionen statt.

Da zur Regelung der Verhaltensplanung auf der überlagerten Basis-Ebene die durch Verhaltensantizipation ermittelte kostenminimale Instruktion zu übergeben

und insgesamt eine Integration in das Schema der hierarchischen Verhaltensplanung (vgl. Abs. 2.2.1.1) zu leisten ist, befasst sich das sechste Arbeitspaket (**AP6**) mit der Konzeption einer Regelung von Verhaltensplanung durch Instruktionen und einer Integration in die hierarchische Verhaltensplanung.

Im letzten Arbeitspaket (**AP7**) sind abschließend die Validierung der Methoden und die Evaluation der Ergebnisse zu leisten.

Die Tabelle 4.1 zeigt eine Zusammenfassung des Handlungsbedarfs bzw. der zu leistenden Arbeit, gruppiert nach den Bereichen Erfassung und Klassifikation, Verhaltensantizipation, Verhaltensregelung sowie Validierung und Evaluation.

5 Konzeption

Ziel dieses Kapitels ist die Konzeption einer Verhaltensantizipation und -regelung kognitiver mechatronischer Systeme bei langfristiger Planung und Ausführung. Die Abbildung 5.1 zeigt zur Übersicht den Ablauf des angestrebten Gesamtverfahrens zu einer repräsentativen Top- und Basis-Ebene der hierarchischen Verhaltensplanung und -regelung (vgl. Abs. 2.2.1). Als Einstiegspunkt des Verfahrens dient die in der Problemstellung thematisierte ereignisorientiert angestoßene rollierende Verhaltensplanung (vgl. Abs. 2.2.2). Zunächst wird mithilfe eines Planers das Verhalten des kognitiven mechatronischen Systems auf der Basis-Ebene für eine vorgegebene Planreichweite eingeplant. Dabei kann z. B. auf das bereits bestehende Verfahren von Klöpper[225] (vgl. auch hybride Planungsarchitektur in Abs. 3.1.1) zurückgegriffen werden. Im Anschluss daran erfolgt die Ausführung des Planes durch das System.

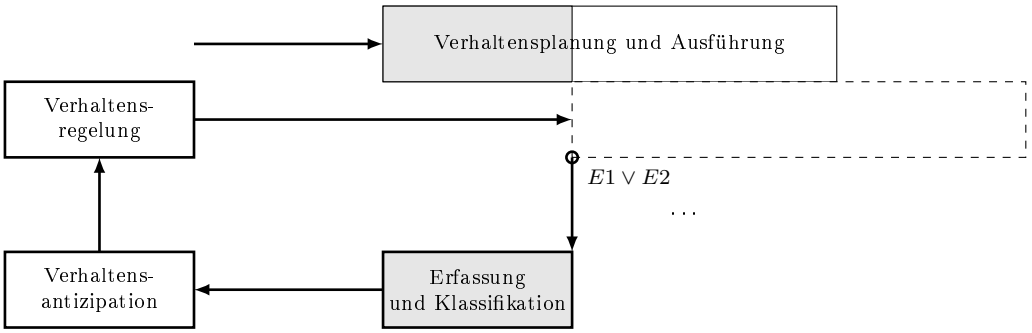


Abbildung 5.1: Gesamtablauf bei einer ereignisorientiert angestoßenen rollierenden Verhaltensplanung

Mit dem Eintreten der Ereignisse E1 oder E2[226] beginnt ein adaptierender Regelkreis, in dem das kognitive mechatronische System folgende Vorgänge hintereinander durchläuft (vgl. Abb. 5.1):

1. Erfassen und Klassifizieren des auf der Basis-Ebene ausgeführten Teilplanes p_i' im prädiktiven Modell auf der Top-Ebene (Erfassung und Klassifikation)
2. Antizipation des kostenminimalen Verhaltens der Basis-Ebene auf Grundlage des prädiktiven Modells auf der Top-Ebene (Verhaltensantizipation)

[225] Vgl. [Klö09], insb. S. 43 ff.

[226] E1: Die vorgegebene maximale Anzahl an auszuführenden Aktionen im Plan p_i ist erreicht; E2: Es existiert für den aktuellen Zustand keine eingeplante Aktion im Plan p_i (vgl. Abs. 2.2.2).

3. Regelung der überlagerten Verhaltensplanung durch Vorgabe des kostenminimalen und als erstes auszuführenden Teilplanes π^* (Verhaltensregelung)
4. Erstellen eines neuen bzw. aktualisierten Planes p_{i+1} unter Einbezug des durch die Top-Ebene vorgegebenen Teilplanes π^* (Verhaltensplanung)
5. Ausführen des neuen bzw. aktualisierten Planes p_{i+1} (Ausführung)
6. Wiederhole Schritt 1 bis 5 ...

In den folgenden Abschnitten 5.1 - 5.3 werden die erforderlichen Teilbereiche zur Realisierung des Verfahrens erarbeitet. Weil eine Datenbasis zur Analyse des langfristigen Verhaltens erforderlich ist, wird in Abschnitt 5.1 zuerst die Erfassung und Klassifikation von ausgeführten Teilplänen entwickelt. Darauf aufbauend folgt in Abschnitt 5.2 die Konzeption einer Verhaltensantizipation, da das Ermitteln des erwarteten zukünftigen Verhaltens eine wesentliche Entscheidungsgrundlage liefert. Anschließend ist in Abschnitt 5.3 die Verhaltensregelung Gegenstand näherer Betrachtung, zumal diese die eigentliche Adaption des Verhaltens unter langfristigen Gesichtspunkten realisiert.

5.1 Erfassung und Klassifikation von Plänen

Die Abb. 5.2 veranschaulicht das Vorgehen zur Erfassung und Klassifikation ausgeführter Teilpläne während der ereignisorientiert angestoßenen rollierenden Planung im Zeitverlauf.

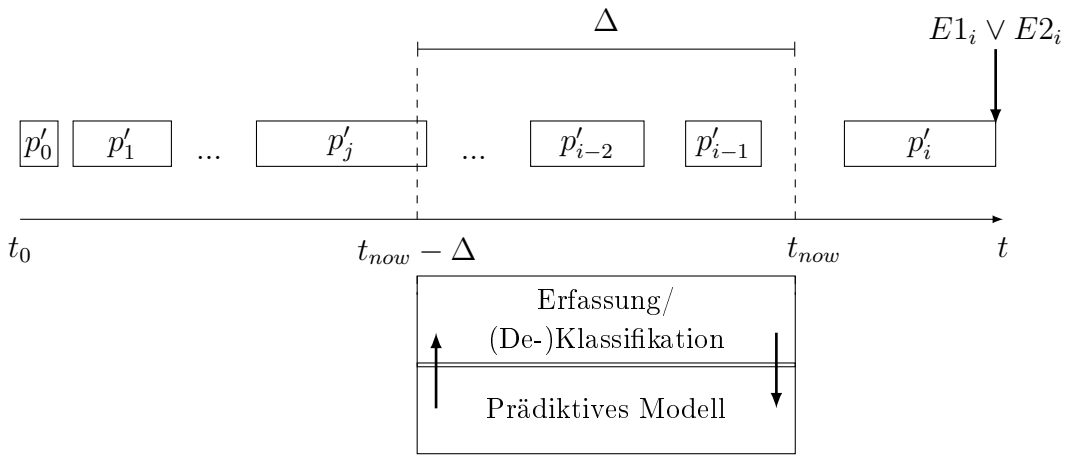


Abbildung 5.2: Erfassung und Klassifikation von Plänen im Zeitverlauf

Ausgehend vom aktuellen Zeitpunkt t_{now} wird während der Ausführung des Planes p_i und nach dem Eintreten des Ereignisses $E1_i$ oder $E2_i$ als Nächstes der bis dahin realisierte Teilplan p'_i zu dem prädiktiven Modell hinzugefügt sowie mittels eines Klassifikators entsprechend seiner Zustandseigenschaften klassifiziert (vgl. Abb. 5.2, p'_i). Teilpläne, die vor dem Zeitpunkt $t_{now} - \Delta$ ausgeführt wurden, also ggf. weit in der Vergangenheit liegen, werden wieder aus dem prädiktiven Modell entfernt

bzw. deklassifiziert (vgl. Abb. 5.2, p'_j). Das prädiktive Modell beinhaltet daher ausschließlich Teilpläne, die im Zeitraum $t_{now} - \Delta$ bis t_{now} ausgeführt wurden.

Durch ein mitlaufendes und von Δ abhängiges Zeitfenster ist zum einen die betrachtete Historie und zum anderen der Einfluss neuer hinzukommender Teilpläne steuerbar. Andernfalls würde ein fortwährendes Erfassen sämtlicher ausgeführter Teilpläne mit der Zeit zu einem prädiktiven Modell führen, bei dem neu hinzukommende Teilpläne keine signifikante Veränderung mehr herbeiführten und so insb. Prognosen hinsichtlich kurzfristiger Entscheidungen eine Art Trägheit aufwiesen.[227]

Die von dem kognitiven mechatronischen System in einem Planungszyklus ausgeführte Folge von Aktionen $a_1, ..., a_m$ ist hier als realisierter Teil eines vorliegenden Planes gegeben, der im Folgenden auch einfach nur als Plan bezeichnet wird (vgl. Abs. 2.2.2). Diese Aktionsfolge wird durch die Alternative

$$L^{actions} = ((a, l^{actions})|nil)^{228} \tag{5.1}$$

modelliert, die eine Liste von Aktionsinstanzen repräsentiert und auf der untersten Basis-Ebene der Gesamthierarchie (vgl. Abs. 2.2.1.1) den ausgeführten Operationsmodi des kognitiven mechatronischen Systems entspricht (vgl. Abs. 2.1.1). Ein ausgeführter Plan p' wird dann zusammen mit seiner Vor- und Nachbedingung (vgl. Abs. 2.2.2) durch

$$P' = (\vec{v}, l^{actions}, \vec{n}) \tag{5.2}$$

modelliert und als abgeschlossene Einheit erfasst. Diese vereinfachte Modellierung soll als Schnittstelle zu beliebigen Planungssystemen dienen. Der Algorithmus 1 zeigt den Ablauf zur Erfassung und Klassifikation eines Planes p'_i bei gegebenem prädiktiven Modell Σ, einer Liste $l^{p'}_{cap}$ von vorangegangenen Plänen und der Zeitdauer Δ.

Mit dieser Ausgangsebene werden in den nächsten Abschnitten das zugehörige prädiktive Modell und die Klassifikation der in einem Zeitraum ausgeführten und erfassten Pläne konzipiert. Dabei ist insb. die Entwicklung der Operationen Hinzufügen (bzw. Klassifizieren) und Entfernen (bzw. Deklassifizieren) von Plänen auf dem prädiktiven Modell von zentraler Bedeutung. Zunächst wird die Anforderung der Klassifikation (vgl. Abs. 2.2.3.2) zurückgestellt und der grundlegende Aufbau des prädiktiven Modells auf Basis eines Markov-Entscheidungsprozesses (vgl. Abs. 3.2.1) konzipiert (vgl. Abs. 5.1.1). Zum Erfüllen der genannten Anforderung folgt dann nachträglich die Erweiterung des Markov-Entscheidungsprozesses durch Zustandsklassen sowie die Umsetzung der Klassenbildung (vgl. Abs. 5.1.3) und Klassifizierung (vgl. Abs. 5.1.3) von Zuständen.

[227] Dies ist in etwa vergleichbar mit dem Gewichtungsfaktor α von Verfahren zur Zeitreihenanalyse, wie beispielsweise der exponentiellen Glättung.

[228] Der rekursive Ausdruck beschreibt die Typisierung einer Liste in Tupel-Schreibweise. Großbuchstaben bezeichnen den Typ und Kleinbuchstaben eine Instanz. Eine Listeninstanz $l^{actions}$ mit drei Aktionen lässt sich beispielsweise als $l^{actions} := (a_0, (a_1, (a_2, (nil))))$ oder in vereinfachter Darstellung als $l^{actions} := (a_0, a_1, a_2)$ aufschreiben.

Algorithm 1: CAPTURE-PLAN

Input: p'_i
Data: Σ, $l^{p'}_{cap}$, Δ
Output: –

```
/* adding/classifying plan p'_i                                   */
classifyPlan(p'_i);
l^{p'}_{cap}.add(p'_i);
/* removing/declassifying oldest plan                             */
while l^{p'}_{cap}.length > 0 and t(l^{p'}_{cap}.get(0).n) < t_now − Δ do
    declassifyPlan(l^{p'}_{cap}.get(0));
    l^{p'}_{cap}.remove(0);
```

5.1.1 Aufbau eines adaptiven Markov-Entscheidungsprozesses

Zur Erfassung der Pläne wird ein prädiktives Modell in Form eines adaptiven MDPs (vgl. Abs. 3.2.1 und Def. 5.1.1) aufgebaut. Dieser dient insb. als Grundlage für die später zu konzipierende Verhaltensantizipation. Der adaptive MDP besteht aus allen im Zeitraum $t_{now} - \Delta$ bis t_{now} vom kognitiven mechatronischen System ausgeführten unterschiedlichen Aktionsfolgen (im Folgenden als eine Menge von Planknoten bezeichnet, vgl. Aktionsmenge in Abs. 3.2.1), den erreichten Zuständen (Zustandsknoten bestehend aus Vor- oder Nachbedingung, vgl. Abs. 2.2.2) und einer Menge von gerichteten Kanten. Die gerichteten Kanten bilden die Vorgänger-/Nachfolgerbeziehung zwischen erreichten Zuständen und ausgeführten Aktionsfolgen im aktuellen Zeitraum ab. Sie sagen aus, dass zum einen mindestens ein Mal eine Aktionsfolge $l_i^{actions}$ ausgehend von einer betrachteten Vorbedingung $\vec{v_k}$ ausgeführt wurde und zum anderen die Ausführung einer Aktionsfolge $l_i^{actions}$ mindestens ein Mal zu einer entsprechenden Nachbedingung $\vec{n_j}$ führte (vgl. Abb. 5.3).

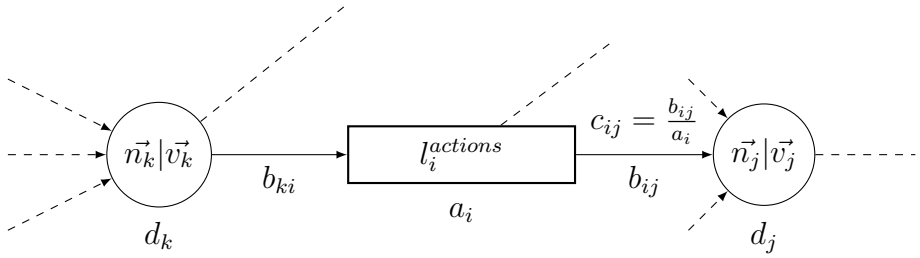


Abbildung 5.3: Allgemeiner Aufbau eines adaptiven MDPs (Ausschnitt)

An jedem Zustands- und Planknoten des adaptiven MDPs wird mit einer Zählvariablen gezählt, wie oft der jeweilige Zustand oder die jeweilige Aktionsfolge $l_i^{actions}$

im Zeitraum erreicht bzw. ausgeführt wurde (vgl. Abb. 5.3, d_k, d_j und a_i). Zusätzlich erfassen weitere Zählvariablen an den Kanten, wie oft die jeweilige Vorgänger-/Nachfolgerbeziehung, ausgehend vom betrachteten Zustands- oder Planknoten, im beobachteten Zeitraum auftauchte (vgl. Abb. 5.3, b_{ki} und b_{ij}).

Definition 5.1.1 (Adaptiver MDP) *Ist ein 3-Tupel $\Sigma = (S, A, P)$ mit Zustandsmenge S, Aktionsmenge $A = \{l_1^{actions}, ..., l_n^{actions}\}$ und einer Menge P von Zählvariablen. Für jeden Zustand $s_k \in S$ sei d_k die Anzahl Zustandserreichungen von s_k, und für jede Aktionsfolge $l_i^{actions} \in A$, a_i die Anzahl Ausführungen von $l_i^{actions}$ im betrachteten Zeitraum δ. Weiterhin sei b_{ki} die Anzahl von Ausführungen der Aktionsfolge $l_i^{actions}$ ausgehend vom Zustand s_k und b_{ij} die Anzahl von anschließenden Zustandserreichungen s_j im betrachteten Zeitraum δ. Gegeben sind die Gewichtungswerte $c_{ij} = b_{ij}/a_i$ mit $c_{ij} = 0$ falls $a_i = 0$.*

Auf Basis dieser Zählvariablen ist zu jeder ausgehenden Kante eines Planknotens ein Gewichtungswert[229] bestimmbar, der das Verhältnis von nachfolgenden Zustandserreichungen zur Anzahl von Aktionsausführungen des betrachteten Zustands- oder Planknotens angibt (vgl. Abb. 5.3, c_{ki} und c_{ij}). Mithilfe der Gewichtungswerte sollen im Rahmen der später in dieser Arbeit zu entwickelnden Verhaltensantizipation eine wesentliche Fragestellung beantwortet werden können: Welcher Folgezustand ist bei einer festgelegten Planausführung auf der überlagerten Basis-Ebene zu erwarten?

Ein adaptiver MDP lässt sich modellieren durch Planknoten, Zustandsknoten, Listen und Kanten. Die Planknoten werden dabei durch das 3-Tupel

$$PN = (l^{actions}, \#a, l^{states}) \tag{5.3}$$

repräsentiert.[230] Jeder Planknoten pn besteht daher aus einer Aktionsfolge $l^{actions}$ (vgl. Ausdruck 5.1), einer Zählvariablen $\#a$ und einer Liste l^{states}. Die Liste l^{states} ist eine Ausprägung der Alternative

$$L^{states} = ((e^{state}, l^{states})|nil) \tag{5.4}$$

und beinhaltet sämtliche ausgehende Kanten des Planknotens. Die Kanten sind Instanzen des 3-Tupels

$$E^{state} = (sn, \#b, c) \tag{5.5}$$

mit dem nachfolgenden Zustandsknoten sn, einer Zählvariablen $\#b$ und einem Gewichtungswert c. Zustandsknoten wiederum werden durch

$$SN = (\vec{s}, \#d, l^{plans}) \tag{5.6}$$

[229] $]0,1] := c \in \mathbb{R} : 0 < c \leq 1$, denn z. B. für jeden Planknoten gilt $\sum_{j=1}^{|E_i|} b_{ij} \leq a_i$ mit E_i als Menge aller ausgehenden Kanten.

[230] # kennzeichnet hier eine Zählvariable.

repräsentiert und kapseln einen Zustandsvektor (eine Vorbedingung $\vec{v}$ oder eine Nachbedingung $\vec{n}$), eine Liste l^{plans} mit ausführbaren Planknoten, modelliert durch die Alternative

$$L^{plans} = ((e^{plan}, l^{plans})|nil), \tag{5.7}$$

sowie eine weitere Zählvariable $\#d$. Die Kante e^{plan} ist eine Instanz des 3-Tupels

$$E^{plan} = (pn, \#b) \tag{5.8}$$

mit einem Planknoten pn als Zielknoten und einer Zählvariablen $\#b$.

Die Abbildung 5.4 zeigt ein Beispiel zum Ausschnitt eines adaptiven MDPs. Dieser umfasst die vier Planknoten pn_{42}, pn_{44}, pn_{45} und pn_{47}, vier Zustandsknoten sn_{41}, sn_{46}, sn_{43} und sn_{44} sowie die durch Planausführungen gebildeten Kanten mit Vorgänger-/Nachfolgerbeziehungen der erfassten Aktionsfolgen bzw. Vor- und Nachbedingungen. An den Zustands- bzw. Planknoten sind jeweils die Zählvariablen $\#d$ und $\#a$ sowie an den Kanten die Zählvariablen $\#b$ und der Gewichtungswert c abgetragen (vgl. Abb. 5.4).

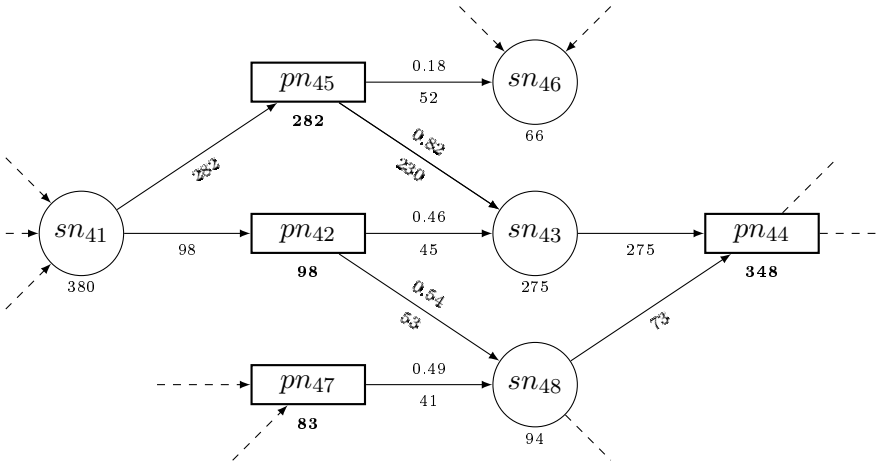


Abbildung 5.4: Beispielausschnitt eines adaptiven MDPs

Beispielsweise wurde auf den Planknoten pn_{45} bereits 282 mal zurückgegriffen und die enthaltene Aktionsfolge $l_{45}^{actions}$ ausgeführt. Wie an der Zählvariablen $\#b$ der Kante von pn_{45} nach sn_{46} abzulesen ist, folgte die im Zustandsknoten sn_{46} enthaltene Nachbedingung (bzw. Vorbedingung) 52 mal nach der Ausführung der im Planknoten pn_{45} enthaltenen Aktionsfolge $l_{45}^{actions}$. Die Division von Zählvariable $\#b = 52$ und Zählvariable $\#a = 282$ ergibt den Gewichtungswert $c = 52/282 = 0.18$ (vgl. Abb. 5.4). Wandert man nun, z. B. ausgehend vom Zustandsknoten sn_{41}, entlang der Kanten mit den höchsten Gewichtungen über Planknoten pn_{45} zum Zustandsknoten sn_{43}, so repräsentiert dieser Pfad eine Sequenz von Vorbedingung, ausgeführter Aktionsfolge und Nachbedingung die von dem kognitiven mechatronischen System in dieser Situation am häufigsten im beobachteten Zeitraum durchlaufen wurde. Eben diese Pfade bilden später die Basis für die Verhaltensantizipation

des Systems. Insb. Aufwandsabschätzungen zum Erreichen bestimmter Ziele sollten unter Berücksichtigung dieser Sequenzen durchgeführt werden. Die zugehörige Modellierung des gesamten Ausschnittes des adaptiven MDPs ist der Tabelle 5.1 zu entnehmen.[231]

Plan-/Zustandsknoten	Listen	Kanten
$pn_i := (l_i^{actions}, \#a_i, l_i^{states})$ $sn_k := (\vec{s}_k, \#d_k, l_k^{plans})$	$l_i^{states} := (e_{ij}^{state}, \dots)$ $l_i^{plans} := (e_{ki}^{plan}, \dots)$	$e_{ij}^{state} := (sn_j, \#b_{ij}, c_{ij})$ $e_{ki}^{plan} := (pn_i, \#b_{ki})$
$sn_{41} := (l_{41}^{actions}, 380, l_{41}^{plans})$	$l_{41}^{plans} := (e_{4145}^{plan}, e_{4142}^{plan})$	$e_{4145}^{plan} := (pn_{45}, 282)$ $e_{4142}^{plan} := (pn_{42}, 98)$
$pn_{42} := (l_{42}^{actions}, 98, l_{42}^{states})$	$l_{42}^{states} := (e_{4243}^{state}, e_{4248}^{state})$	$e_{4243}^{state} := (sn_{43}, 45, 0.46)$ $e_{4248}^{state} := (sn_{48}, 53, 0.54)$
$sn_{43} := (l_{43}^{actions}, 275, l_{43}^{plans})$	$l_{43}^{plans} := (e_{4344}^{plan})$	$e_{4344}^{plan} := (pn_{44}, 275)$
$pn_{44} := (l_{44}^{actions}, 348, l_{44}^{states})$	$l_{44}^{states} := (...)$	
$pn_{45} := (l_{45}^{actions}, 282, l_{45}^{states})$	$l_{45}^{states} := (e_{4546}^{state}, e_{4543}^{state})$	$e_{4546}^{state} := (sn_{46}, 52, 0.18)$ $e_{4543}^{state} := (sn_{43}, 230, 0.82)$
$sn_{46} := (l_{46}^{actions}, 66, l_{46}^{plans})$	$l_{46}^{plans} := (...)$	
$pn_{47} := (l_{47}^{actions}, 83, l_{47}^{states})$	$l_{47}^{states} := (e_{4748}^{state})$	$e_{4748}^{state} := (sn_{48}, 41, 0.49)$
$sn_{48} := (l_{48}^{actions}, 94, l_{48}^{plans})$	$l_{48}^{plans} := (e_{4844}^{plan})$	$e_{4844}^{plan} := (pn_{44}, 73)$

Tabelle 5.1: Modellierung eines adaptiven MDPs – Beispiel

Es bleibt noch zu erwähnen, dass ein adaptiver MDP in den meisten Fällen von der Anzahl her weit weniger Zustands- und Planknoten enthält, als im betrachteten Zeitraum δ Zustände (Vor- oder Nachbedingungen) und Aktionsfolgen erfasst wurden. Nur unterschiedliche Zustände und Pläne führen zu neuen Knoten im adaptiven MDP, während bereits vorhandene Zustände und Pläne lediglich eine Erhöhung der Zählvariablen zur Folge haben. In den nächsten Abschnitten wird gezeigt, wie die erfassten Pläne mit ihren Vor- und Nachbedingungen (vgl. Ausdruck 5.2) zum adaptiven MDP hinzugefügt bzw. wieder entfernt werden.

Weil vor diesen Operationen jeweils zu überprüfen ist, ob die Vorbedingung, die Aktionsfolge und die Nachbedingung bereits in identischer Form im adaptiven MDP vorliegen, wird zunächst die Suche nach diesen identischen Zuständen und Plänen

[231] Bemerkung: Die Verschachtelung der Klammern durch das rekursive Aufbauen der Listen sowie die nil-Ausdrücke wurden der Übersicht halber nicht abgebildet.

behandelt. Dies ist erforderlich, um zu unterscheiden, ob lediglich eine Aktualisierung der Zählvariablen, oder ggf. eine Erweiterung bzw. Reduzierung der Struktur des adaptiven MDPs um Zustandsknoten oder Planknoten zu erfolgen hat.

5.1.1.1 Suchen von Plänen

Ziel ist es, zu einem gegebenen Plan p'_i (vgl. Ausdruck 5.2) zwei, mit einer Aktionsfolge $l_s^{actions}$ (vgl. Ausdruck 5.1) durch Kanten direkt verbundene, Zustände $\vec{v}_s$ und $\vec{n}_s$ im adaptiven MDP zu finden, die einen Plan $p'_s = (\vec{v}_s, l_s^{actions}, \vec{n}_s)$ repräsentieren (Planrepräsentation) für den $p'_s = p'_i$ gilt. Für diese Gleichheit ist zunächst ein Verständnis zu entwickeln und zu klären, wann zwei Pläne wirklich identisch sind. Bei der Gleichheit von Plänen wird hier zwischen zwei Arten der Übereinstimmung unterschieden: Der schwachen und der starken Plangleichheit. Die schwache Plangleichheit (vgl. Def. 5.1.2) liegt vor, wenn die zwei zu vergleichenden Pläne p'_s und p'_i mindestens in ihren Vor- sowie Nachbedingungen übereinstimmen, also $\vec{v}_s = \vec{v}_i$ und $\vec{n}_s = \vec{n}_i$ gilt.[232] Daraus folgt, dass die Ausführung beider Pläne mit der ersten Aktion im gleichen Ausgangszustand gestartet wurde und im Anschluss mit der letzten Aktion zum gleichen Zielzustand führte. Dabei enthalten die Pläne ggf. eine voneinander abweichende Anzahl an völlig unterschiedlichen Aktionen und lösen dadurch die gleiche Aufgabe auf verschiedene Art und Weise. Die schwache Plangleichheit ist daher für das Finden einer identischen Planrepräsentation nicht ausreichend, liefert aber eine wesentliche Information über das Verhältnis von Plänen.

Definition 5.1.2 (schwache Plangleichheit) *Zwei Pläne p'_1 und p'_2 sind schwach vergleichbar ($p'_1 \approx p'_2$), wenn $\vec{v}_{p'_1} = \vec{v}_{p'_2} \wedge \vec{n}_{p'_1} = \vec{n}_{p'_2}$ gilt.*

Eine starke Plangleichheit (vgl. Def. 5.1.3) ist gegeben, wenn die zwei betrachteten Pläne p'_s und p'_i nicht nur in ihren Vor- und Nachbedingungen, sondern auch zusätzlich bzgl. ihrer Aktionsfolge völlig identisch sind. Um Letzteres zu gewährleisten, müssen zum einen die Aktionsfolgen $l_s^{actions}$ und $l_i^{actions}$ die gleiche Anzahl an Aktionen beinhalten ($|l_s^{actions}| = |l_s^{actions}|$) und zum anderen alle Aktionen $a_r \in l_s^{actions}$ und $a_j \in l_i^{actions}$ mit $r = j$ paarweise übereinstimmen ($a_r = a_j$). Ist dies zusätzlich zur schwachen Plangleichheit für sämtliche Aktionen erfüllt, so ist auch die starke Plangleichheit erfüllt.

Definition 5.1.3 (starke Plangleichheit) *Zwei Pläne p'_1 und p'_2 sind stark vergleichbar ($p'_1 = p'_2$), wenn sie nach Definition 5.1.2 schwach vergleichbar sind und zusätzlich $|l_1^{actions}| = |l_2^{actions}|$ sowie $\forall a_i \in l_1^{actions}, a_j \in l_2^{actions} : i = j \rightarrow a_i = a_j$ erfüllt ist.*

[232] Hinweis: Die Vorbedingung eines Planes p'_i entspricht der Vorbedingung der ersten in diesem Plan ausgeführten Aktion a_1 und damit gilt: $\vec{v}_{p'_i} = \vec{v}_{a_1}$ mit $a_1 \in p'_i$. Analog entspricht die Nachbedingung eines Planes p'_i der Nachbedingung der letzten ausgeführten Aktion: $\vec{n}_{p'_i} = \vec{n}_{a_{|p'_i|}}$ mit $a_{|p'_i|} \in p'_i$ und $|p'_i|$ für die Anzahl an ausgeführten Aktionen im Plan p'_i.

Die Suche im adaptiven MDP nach einer zu p'_i vergleichbaren Planrepräsentation p'_s kann nun unter Verwendung dieser zwei zuvor beschriebenen Kriterien zur Plangleichheit durchgeführt werden. Ist beispielsweise das Finden von alternativen Ausführungsverläufen erforderlich, so wird nach Planrepräsentationen gesucht, die schwach vergleichbar sind. Eine weitere Steigerung besteht z. B. darin, den gegebenen Plan und die gefundene Planrepräsentation hinsichtlich ihres prozentualen Anteils an paarweise übereinstimmenden Aktionen miteinander zu vergleichen. Soll ausschließlich überprüft werden, ob ein betrachteter Plan in identischer Form in dem adaptiven MDP existiert, so wird nach einer Planrepräsentation gesucht, die stark vergleichbar ist.

Der Algorithmus 2 zeigt den Ablauf der Suche eines zum Zustand $\vec{s}_i$ zugehörigen Zustandsknoten sn_s in einer gegebenen Zustandsmenge S eines adaptiven MDPs Σ. Der Algorithmus 3 zeigt den Ablauf der Überprüfung von zwei gegebenen Aktionsfolgen $l_1^{actions}$ und $l_2^{actions}$ auf ihre Gleichheit. Der Algorithmus 4 zeigt den Ablauf der Suche eines Planknotens pn_s in einem MPD Σ bei gegebener Aktionsfolge $l^{actions}$.

Algorithm 2: SEARCH-STATENODE

Input: $\vec{s}_i$
Data: Σ
Output: sn_s

```
1 foreach sn_s of Σ.S do
2     if sn_s.s⃗ = s⃗_i then
3         return sn_s
4 return nil;
```

Algorithm 3: IS-EQUAL

Input: $l_1^{actions}, l_2^{actions}$
Data: -
Output: true/false

```
1 if |l_1^actions| ≠ |l_2^actions| then
2     return false
3 for m = 1 to |l_2^actions| do
4     if l_1^actions.a_m ≠ l_2^actions.a_m then
5         return false
6 return true;
```

Nach diesen Vorüberlegungen können nun in den nächsten zwei Abschnitten die Operationen Hinzufügen und Entfernen von Plänen zum adaptiven MDP konzipiert werden.

Algorithm 4: SEARCH-PLANNODE

Input: $l^{actions}$
Data: Σ
Output: pn_s

```
foreach pn_s of Σ do
    if isEqual(l^actions, pn_s.l^actions) then
        return pn_s
return nil;
```

5.1.1.2 Hinzufügen von Plänen

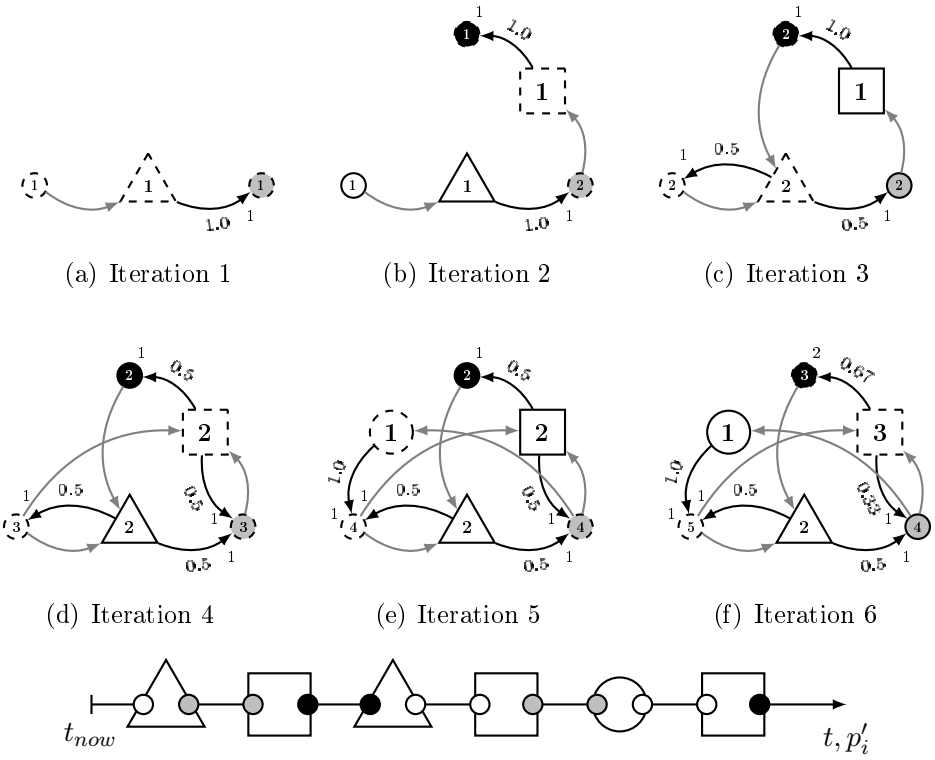
Die Operation Hinzufügen eines neuen Planes p'_i zu einem adaptiven MDP (vgl. Def. 5.1.1) besteht aus vier aufeinander aufbauenden Phasen:

1. Suchen von Zustands- und Planknoten
2. Heraufsetzen und Erzeugen von Zustands- und Planknoten
3. Heraufsetzen und Bilden von Kanten zu Zustands- und Planknoten
4. Aktualisieren der Gewichtungswerte von Kanten zu Zustands- und Planknoten

In der ersten Phase wird zum gegebenen Plan $p'_i = (\vec{v}_i, l_i^{actions}, \vec{n}_i)$, im adaptiven MDP, zum einen nach zwei Zustandsknoten sn_{pre} und sn_{post}, mit $sn_{pre}.\vec{s} = \vec{v}_i$ und $sn_{post}.\vec{s} = \vec{n}_i$, und zum anderen nach einem Planknoten pn_s mit $pn_s.l^{actions} = l_i^{actions}$ gesucht. Ist diese Suche für Vorbedingung, Aktionsfolge und Nachbedingung des Planes p'_i erfolgreich, so handelt es sich um eine bereits im MDP vorhandene stark vergleichbare Planrepräsentation mit $p'_s = (sn_{pre}.\vec{s}, pn_s.l_s^{actions}, sn_{post}.\vec{s}) = p'_i$ (vgl. Def. 5.1.3). Somit ist kein neuer Plan- oder Zustandsknoten im adaptiven MDP zu generieren, sondern lediglich die Zählvariablen bestehender Knoten anzupassen und ggf. neue Kanten zu bilden (vgl. Abb. 5.5 und Alg. 5).

Die zweite Phase umfasst das Heraufsetzen bereits bestehender bzw. das Generieren von neuen Zustands- und Planknoten. Bei einer erfolglosen Suche nach den Zustandsknoten zur Vor- bzw. Nachbedingung wird ein neuer Zustandsknoten sn_{pre} bzw. sn_{post} generiert und mit Zählvariable $\#d = 1$ initialisiert. Die Zählvariable $\#d$ bereits existierender Zustandsknotens für die Vor- oder Nachbedingung wird hingegen um den Wert 1 erhöht. Ist die Suche nach dem Planknoten nicht erfolgreich, so wird ein neuer Planknoten pn_s mit $\#a = 1$ neu erzeugt, andernfalls die Zählvariable a um den Wert 1 erhöht (vgl. Abb. 5.5 und Alg. 5).

Die Aufgabe der dritten Phase besteht im Heraufsetzen bereits bestehender bzw. dem Generieren neuer Kanten. Es findet eine Überprüfung statt, ob zur Vorgänger-/Nachfolgebeziehung von Vorbedingung, Aktionsfolge und Nachbedingung des Planes p'_i bereits Kanten von sn_{pre} nach pn_s und von pn_s nach sn_{post} gegeben sind. Anschließend folgt entsprechend das Bilden nicht-existenter Kanten, mit initialer Zählvariable $b = 1$, sowie das Erhöhen der Zählvariablen bereits bestehender Kanten um den Wert 1 (vgl. Abb. 5.5 und Alg. 5).



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

(d) Iteration 4 (e) Iteration 5 (f) Iteration 6

(g) Sequenz mit 3 verschiedenen Plänen und Zuständen im Zeitverlauf

Abbildung 5.5: Adaptiver MDP - Hinzufügen von Plänen

Weil das Heraufsetzen von Zustandsknoten, Planknoten oder Kanten eine Beeinflussung der Zählvariablen zur Folge hat, ist eine anschließende Aktualisierung der Gewichtungswerte aller ausgehenden Kanten des Zustandsknotens sn_{pre} und des Planknotens pn_s unerlässlich. Diese Aufgabe ist der vierten Phase zugeordnet.

Die Abbildung 5.5 zeigt beispielhaft das Vorgehen zum Hinzufügen von neuen Plänen zu einem adaptiven MDP im Zeitverlauf (Zählvariablen von ausgehenden Kanten der Zustandsknoten sind der Übersicht halber nicht abgebildet). Der Algorithmus 5 zeigt den konkreten Ablauf zum Hinzufügen eines Planes p'_i zu einem gegebenen adaptiven MDP Σ.

In diesem Abschnitt wurde gezeigt, wie neue Pläne zu einem adaptiven MDP im Zeitverlauf hinzugefügt werden können. Um ein ggf. mitlaufendes Zeitfenster zu realisieren, ist zusätzlich das Entfernen von Plänen zu behandeln.

Algorithm 5: ADD-PLAN-TO-MDP

Input: p'_i
Data: Σ
Output: -

`/* searching for state and plan nodes */`
$sn_{pre} \longleftarrow \Sigma$.`searchStateNode`$(p'_i.\vec{v})$
$pn_s \longleftarrow \Sigma$.`searchPlanNode`$(p'_i.l^{actions})$
$sn_{post} \longleftarrow \Sigma$.`searchStateNode`$(p'_i.\vec{n})$
`/* updating and generating state and plan nodes */`
if $sn_{pre} = nil$ **then**
  $sn_{pre} \longleftarrow (p'_i.\vec{v}, 1, ())$
  Σ.`addState`(sn_{pre})
else
  $sn_{pre}.\#d \longleftarrow sn_{pre}.\#d + 1$
if $sn_{post} = nil$ **then**
  $sn_{post} \longleftarrow (p'_i.\vec{n}, 1, ())$
  Σ.`addState`(sn_{post})
else
  $sn_{post}.\#d \longleftarrow sn_{post}.\#d + 1$
if $pn_s = nil$ **then**
  $pn_s \longleftarrow (l_i^{actions}, 1, ())$
  Σ.`addPlanNode`(pn_s)
else
  $pn_s.\#a \longleftarrow pn_s.\#a + 1$
`/* updating and generating edges to state and plan nodes */`
if $e_{pre,s} = nil$ **then**
  $e_{pre,s} \longleftarrow (pn_s, 1)$
  $sn_{pre}.l^{plans}$.`add`$(e_{pre,s})$
else
  $e_{pre,s}.\#b \longleftarrow e_{pre,s}.\#b + 1$
if $e_{s,post} = nil$ **then**
  $e_{s,post} \longleftarrow (sn_{post}, 1, 1)$
  $pn_s.l^{states}$.`add`$(e_{s,post})$
else
  $e_{s,post}.\#b \longleftarrow e_{s,post}.\#b + 1$
`/* updating the c-values of edges to state nodes */`
foreach $e_{s,j}$ *of* l_s^{states} **do**
  $e_{s,j}.c \longleftarrow \frac{e_{s,j}.\#b}{pn_s.\#a}$

5.1.1.3 Entfernen von Plänen

Die Operation Entfernen eines Planes p'_i aus einem adaptiven MDP (vgl. Def. 5.1.1) besteht ebenfalls aus vier aufeinander aufbauenden Phasen:

1. Suchen von Zustands- und Planknoten
2. Herabsetzen und Entfernen von Zustands- und Planknoten
3. Herabsetzen und Entfernen von Kanten zu Zustands- und Planknoten
4. Aktualisieren der Gewichtungswerte von Kanten zu Zustands- und Planknoten

Diese können in analoger Weise als symmetrisch zu den Phasen beim Hinzufügen eines Planes (vgl. Abs. 5.1.1.2) verstanden werden.

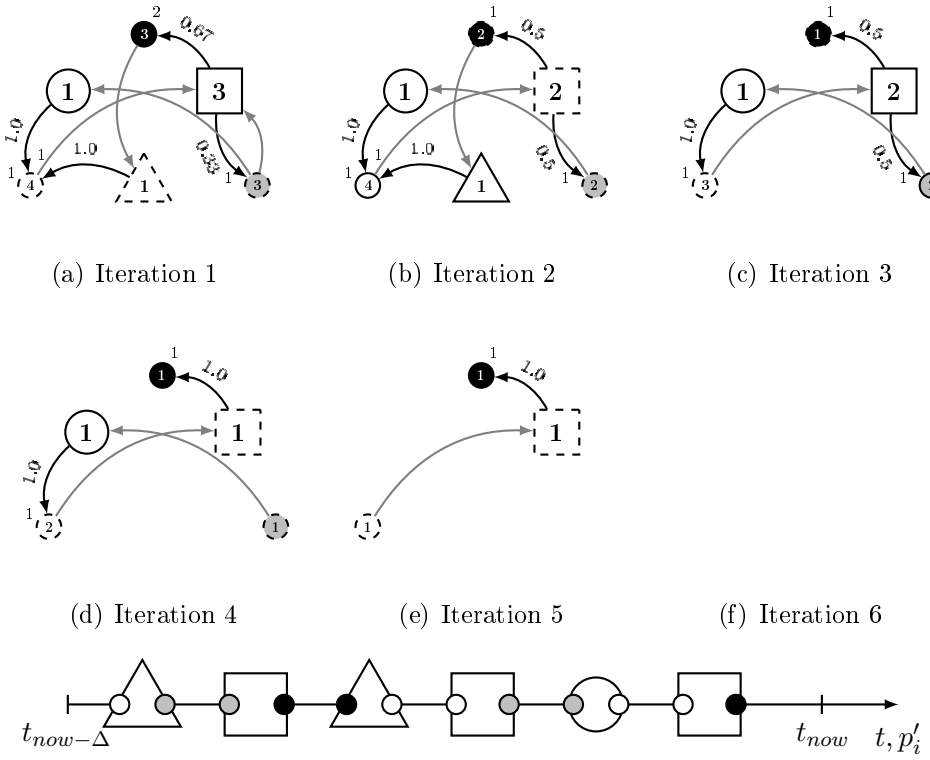
In der ersten Phase wird zum gegebenen Plan $p'_i = (\vec{v}_i, l_i^{actions}, \vec{n}_i)$, im adaptiven MDP, zum einen nach zwei Zustandsknoten sn_{pre} und sn_{post}, mit $sn_{pre}.\vec{s} = \vec{v}_i$ und $sn_{post}.\vec{s} = \vec{n}_i$, und zum anderen nach einem Planknoten pn_s mit $pn_s.l^{actions} = l_i^{actions}$ gesucht. Weil davon auszugehen ist, dass nur ein Plan aus dem adaptiven MDP entfernt wird, wenn in diesem auch eine entsprechende Planrepräsentation existiert, so ist diese Suche für Vorbedingung, Aktionsfolge und Nachbedingung des Planes p'_i in der Regel erfolgreich. Daher ist bereits im adaptiven MDP eine stark vergleichbare Planrepräsentation mit $p'_s = (sn_{pre}.\vec{s}, pn_s.l_s^{actions}, sn_{post}.\vec{s}) = p'_i$ (vgl. Def. 5.1.3) vorhanden.

Die zweite Phase umfasst das Herabsetzen bzw. das Entfernen von Zustands- und Planknoten. Bei einer Zählvariablen $\#d = 1$ wird der Zustandsknoten sn_{pre} bzw. sn_{post} zur Vor- bzw. Nachbedingung entfernt. Die Zählvariable $\#d$ eines bestehen bleibenden Zustandsknotens für die Vor- oder Nachbedingung wird hingegen um den Wert 1 erniedrigt. Ist die Zählvariable des gefundenen Planknotens pn_s mit $a = 1$ gegeben, so wird dieser entfernt, andernfalls wird die Zählvariable a um den Wert 1 erniedrigt (vgl. Abb. 5.6 und Alg. 6).

Die Aufgabe der dritten Phase besteht aus dem Herabsetzen bzw. Entfernen von Kanten. Zur Vorgänger-/Nachfolgebeziehung von Vorbedingung, Aktionsfolge und Nachbedingung des Planes p'_i findet eine Überprüfung der Zählvariablen bestehender Kanten von sn_{pre} nach pn_s und von pn_s nach sn_{post} statt. Anschließend folgt das Entfernen von Kanten mit Zählvariable $b = 1$ sowie das Erniedrigen der Zählvariablen von bestehen bleibenden Kanten um den Wert 1 (vgl. Abb. 5.6 und Alg. 6).

Weil das Herabsetzen von Zustandsknoten, Planknoten oder Kanten eine Beeinflussung der Zählvariablen zur Folge hat, ist eine anschließende Aktualisierung der Gewichtungswerte aller ausgehenden Kanten des Zustandsknotens sn_{pre} und des Planknotens pn_s unerlässlich. Diese Aufgabe ist der vierten Phase zugeordnet.

Die Abbildung 5.6 zeigt beispielhaft das Vorgehen zum Entfernen von Plänen aus dem adaptiven MDP im Zeitverlauf (Zählvariablen von ausgehenden Kanten der Zustandsknoten sind der Übersicht halber nicht abgebildet). Dabei wurde vom zuvor in Abs. 5.1.1.2 aufgebauten adaptiven MDP ausgegangen (vgl. Abb. 5.5). Der Algorithmus 6 zeigt den konkreten Ablauf zum Entfernen eines Planes p'_i aus einem gegebenen adaptiven MDP Σ.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

(d) Iteration 4 (e) Iteration 5 (f) Iteration 6

(g) Sequenz mit 3 verschiedenen Plänen und Zuständen im Zeitverlauf

Abbildung 5.6: Adaptiver MDP - Entfernen von Plänen

In den vorangegangenen Abschnitten wurde gezeigt, wie die während der ereignisorientiert angestoßenen rollierenden Verhaltensplanung ausgeführten Pläne (vgl. Abs. 2.2.2) unter Verwendung des adaptiven MDPs erfasst werden. Durch das gezielte Hinzufügen neuer und Entfernen alter Pläne wurde die Möglichkeit zur Betrachtung eines gegebenen Zeitraums realisiert. Wie sich unmittelbar vermuten lässt, liefert das gezeigte Verfahren zwar eine geeignete Grundlage zum Aufbau eines prädiktiven Modells auf der Top-Ebene (vgl. Abs. 2.2.1.1), ist aber für kognitive mechatronische Systeme in kontinuierlichem Umfeld mit (quasi-)kontinuierlichen Zustandswerten in dieser Form noch nicht ausreichend.

Weil die Vor- und Nachbedingungen der erfassten Pläne in der Regel infinitesimale Unterschiede aufweisen, wird der aufgebaute adaptive MDP viele Zustandsknoten und wenig Wiederholungen aufweisen. Es besteht der Bedarf nach einer Zusammenfassung von Zuständen mit ähnlichen Eigenschaften. Deshalb wird im folgenden Abschnitt das Verfahren um eine Klassifikation von Zuständen erweitert.

Algorithm 6: REMOVE-PLAN-FROM-MDP

Input: p'_i
Data: Σ
Output: -

`/* searching for state and plan nodes */`

$sn_{pre} \longleftarrow \Sigma$.`searchStateNode`$(p'_i.\vec{v})$
$pn_s \longleftarrow \Sigma$.`searchPlanNode`$(p'_i, l_i^{actions})$
$sn_{post} \longleftarrow \Sigma$.`searchStateNode`$(p'_i.\vec{n})$

`/* updating and removing state and plan nodes */`

if $sn_{pre}.\#d = 1$ **then**
  Σ.`removeStateNode`(sn_{pre})
else
  $sn_{pre}.\#d \longleftarrow sn_{pre}.\#d - 1$
if $sn_{post}.\#d = 1$ **then**
  Σ.`removeStateNode`(sn_{post})
else
  $sn_{post}.\#d \longleftarrow sn_{post}.\#d - 1$
if $pn_s.\#a = 1$ **then**
  Σ.`removePlanNode`(pn_s)
else
  $pn_s.\#a \longleftarrow pn_s.\#a - 1$

`/* updating and removing edges to state and plan nodes */`

if $e_{pre,s}.\#b = 1$ **then**
  $sn_{pre}.l^{plans}$.`remove`$(e_{pre,s})$
else
  $e_{pre,s}.\#b \longleftarrow e_{pre,s}.\#b - 1$
if $e_{s,post}.\#b = 1$ **then**
  $pn_s.l^{states}$.`remove`$(e_{s,post})$
else
  $e_{s,post}.\#b \longleftarrow e_{s,post}.\#b - 1$

`/* updating the c-values of edges to state nodes */`

foreach $e_{s,j}$ *of* l_s^{states} **do**
  $e_{s,j}.c \longleftarrow \frac{e_{s,j}.\#b}{pn_s.\#a}$

5.1.2 Klassenbildung von Zuständen

Die kontinuierlichen Prozesse des Systemumfelds führen während der Ausführung des kognitiven mechatronischen Systems zu einer Vielzahl von zu erfassenden Plänen, die sich ggf. nur geringfügig in ihren Vor- sowie Nachbedingungen unterscheiden (vgl. Abs. 2.2.3.2). Weil die Vor- und Nachbedingungen als Zustandsvektoren $\vec{v}_{p_i}, \vec{n}_{p_i} \in \mathbb{R}^n$ vorliegen, existieren grundsätzlich drei Kriterien zur Klassenbildung von Plänen: Vorbedingung, Nachbedingung, Vor- und Nachbedingung.

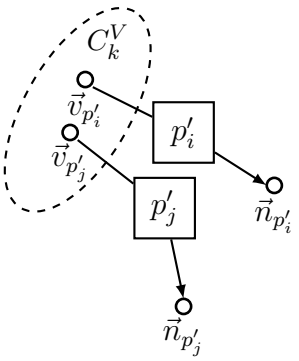


Abbildung 5.7: Klassenbildung von Plänen nach Vorbedingung

Mit der Klassenbildung C^V von Plänen nach Vorbedingungen wird das Ziel verfolgt, bei gegebenem Ausgangszustand $\vec{v}_i$, ähnliche Ausgangszustände zu finden, und die dadurch realisierbaren alternativen Pläne bzw. deren Zielzustände bestimmen zu können.[233] Als Nebenbedingung ergibt sich die Erreichbarkeit der gefundenen Ausgangszustände der alternativen Pläne mit vertretbarem Aufwand. Eine Klassenbildung von Plänen nach Vorbedingungen liefert also eine Übersicht über erreichbare Ausgangszustände, die aber unter Umständen mit der Planausführung zu völlig verschiedenen Zielzuständen führen (vgl. Abb. 5.7).

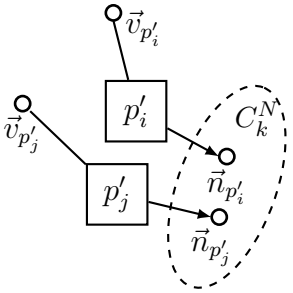


Abbildung 5.8: Klassenbildung von Plänen nach Nachbedingung

Im Gegensatz dazu verfolgt die Klassenbildung C^N von Plänen nach Nachbedingungen das Ziel, bei gegebenem Zielzustand $\vec{n}_i$, ähnliche Zielzustände zu finden und

[233] Beispiel: Wenn der Energiestand des Transportsystems um nur 1,5% aufgeladen wird, dann können zwei weitere Strecken befahren und damit alternativ die Aufträge B und C erfüllt werden.

die dazu erforderlichen alternativen Pläne bzw. Ausgangszustände zu bestimmen.[234] Als Nebenbedingung ergibt sich die Ergebnisgleichheit oder -verbesserung der gefundenen Zielzustände der alternativen Pläne. Eine Klassenbildung nach Nachbedingungen liefert also eine Übersicht über erreichbare Zielzustände von Plänen, die aber unter Umständen mit der Planausführung in völlig verschiedenen Ausgangszuständen starten (vgl. Abb. 5.8).

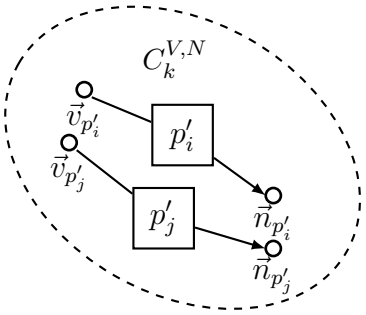


Abbildung 5.9: Klassenbildung von Plänen nach Vor- und Nachbedingung

Die Klassenbildung $C^{V,N}$ von Plänen nach Vor- und Nachbedingungen hat das Ziel, bei gegebenem Ausgangs- und Zielzustand $\vec{v}_i, \vec{n}_i$, ähnliche Ausgangs- und Zielzustände zu finden und die dazu alternativen Pläne bestimmen zu können.[235] Als Nebenbedingung ergibt sich die Erreichbarkeit der gefundenen Ausgangszustände und die Ergebnisgleichheit oder -verbesserung der Zielzustände der Pläne. Eine Klassenbildung von Plänen nach Vor- und Nachbedingungen liefert also eine Übersicht über Pläne, die in ähnlichen Ausgangszuständen mit der Planausführung starten und dabei ähnliche Zielzustände erreichen (vgl. Abb. 5.9).

Zur Klassifikation von Zuständen genügt es, sich in den folgenden Abschnitten ausschließlich auf die Klassenbildung von Vor- oder Nachbedingungen, also auf die Klassenbildung von Systemzuständen des kognitiven mechatronischen Systems, zu konzentrieren. Um Klassen von Systemzuständen in das prädiktive Modell auf der Top-Ebene mit aufzunehmen, wird zunächst das bestehende Modell des adaptiven MDPs entsprechend durch Zustandsklassen angepasst bzw. erweitert (vgl. Abs. 5.1.2.1). Weil die Ähnlichkeit der Zustände während der Klassenbildung zu quantifizieren ist, findet im Anschluss das Entwickeln eines spezialisierten Proximitätsmaßes statt (vgl. Abs. 5.1.2.2). Zum Bilden der Klassen wird schließlich ein partitionierendes Clusteringverfahren auf Basis des *k-Means*-Algorithmus in angepasster Form umgesetzt (vgl. Abs. 5.1.2.3).

[234] Beispiel: Das Transportsystem kann den Auftrag A mit einem um 1,5% höheren Energiestand abschließen, wenn die Auftragsausführung alternativ in den Stationen 2 oder 3 gestartet wird.

[235] Beispiel: Das Transportsystem kann den Auftrag A mit einem 1,5% niedrigeren Energiestand beginnen und mit einem 1,5% höheren Energiestand abschließen, wenn statt Strecke 1 die Strecke 2 befahren wird.

5.1.2.1 Erweiterter adaptiver Markov-Entscheidungsprozess

In diesem Abschnitt wird der bereits konzipierte adaptive MDP (vgl. Abs. 5.1.1) um Zustandsklassen angepasst bzw. erweitert. Ziel ist es, den grundsätzlichen Aufbau der Datenstruktur nach Definition 5.1.1 weitestgehend zu erhalten. Der Ansatz zur Erweiterung besteht darin, einen Zustandsknoten nicht mehr nur als Repräsentant eines einzelnen Systemzustands, sondern als eine Klasse von Systemzuständen mit sehr ähnlichen Zustandswerten zu verstehen. Der resultierende (erweiterte) adaptive MDP besteht daher aus Zustandsklassen, Planknoten und Kanten mit den bereits eingeführten Zählvariablen (vgl. Abs. 5.1.1).

Instanzen von Klassen werden durch das Tupel

$$C = (\vec{m}, l^{conditions}) \tag{5.9}$$

repräsentiert. Diese beinhalten einen Klassenschwerpunkt sowie eine Liste von Zustandsvektoren, modelliert durch die Alternative

$$L^{coditions} = ((\vec{s}, l^{conditions})|nil). \tag{5.10}$$

Der Schwerpunkt $\vec{m}$ repräsentiert alle in der Klasse enthaltenen Vor- bzw. Nachbedingungen und nimmt hier stellvertretend die Rolle des Systemzustands ein. Im (erweiterten) adaptiven MDP kapselt jetzt jeder Zustandsknoten, neben der Zählvariablen und der bereits bekannten Liste von erreichbaren Planknoten (vgl. Ausdruck 5.7), nicht mehr nur einen Zustandsvektor, sondern eine Klasse von Zustandsvektoren. Die Modellierung der Zustandsklassenknoten erfolgt durch das 3-Tupel

$$SN^C = (C, \#d, l^{plans}). \tag{5.11}$$

Weil auch die ausgehenden Kanten von Planknoten in diesem Modell nicht mehr auf Zustandsknoten, sondern auf Zustandsklassenknoten zeigen, ist eine Anpassung des korrespondierenden Ausdruckes (vgl. Ausdruck 5.5) mit

$$E^{state} = (SN^C, \#b, c) \tag{5.12}$$

erforderlich.

Auf Grundlage dieses (erweiterten) adaptiven MDPs können nun die vorbereitenden Maßnahmen zur Durchführung der Klassenbildung auf Basis einer partitionierenden Clusteranalyse getätigt werden.

5.1.2.2 Proximität von Zuständen

Dieser Abschnitt beinhaltet die Herleitung des spezialisierten Proximitätsmaßes zur Quantifizierung der (Un-)Ähnlichkeit von Systemzuständen auf Basis der direkten Effekte von Aktionen. Eine Aktion a überführt eine Vorbedingung $\vec{v}_a$ in eine Nachbedingung $\vec{n}_a$ mit $\vec{v}_a, \vec{n}_a \in \mathbb{R}^n$ (vgl. Abs. 2.2.2). Die Differenz dieser beiden Zustandsvektoren bietet einen Ansatzpunkt zum Ermitteln der direkten Effekte

einer Aktion a. Dabei ist Folgendes zu beachten: Die Komponenten der Vektoren $\vec{v}_a$ und $\vec{n}_a$ sind zwar Werte aus dem reellen Zahlenraum, repräsentieren jedoch ursprünglich Größen aus völlig unterschiedlichen Dimensionen $dim_1, ..., dim_n$ (vgl. Abs. 2.2.3.2).[236]

Weil für die Zustandsvektoren $\vec{v}_a = (v_a(1), ..., v_a(n))$ und $\vec{n}_a = (n_a(1), ..., n_a(n))$ die Eigenschaft $v_a(j), n_a(j) \in dim_j$ mit $1 \leq j \leq n$ erfüllt ist, kann für eine spezifische Komponente die paarweise Differenz $n_a(j) - v_a(j)$ als Effekt der zugehörigen Aktion a in der entsprechenden Dimensionen j aufgefasst werden. Zur Normierung der Differenzwerte in einen einheitlichen Wertebereich wird zu jeder Dimension j ein adäquater Faktor u_j im Vorfeld definiert. Es ergibt sich der Ausdruck

$$u_j(n_a(j) - v_a(j)). \tag{5.13}$$

Darüber hinaus kann das Ergebnis dieser Differenz eine erhöhende oder verringernde Wirkung auf den Effekt ausdrücken. Es ist eine Funktion erforderlich, die diesen Zusammenhang für die jeweilige Dimension abbildet. Die Funktion f_j steht hier stellvertretend für drei alternativ anzuwendende und bzgl. der Dimension auszuwählende Funktionen:

- Die Identitätsfunktion $id(x)$,
- die negierte Identitätsfunktion $-id(x)$
- und die Betragsfunktion $abs(x)$.

Die Funktion $id(x)$ wird gewählt, wenn das positive Ergebnis der Differenz eine erhöhende Wirkung auf den Effekt ausdrückt.[237] Stellt hingegen das negative Ergebnis eine erhöhende Wirkung dar, so wird die Funktion $-id(x)$ verwendet.[238] Repräsentieren sowohl das negative als auch das positive Ergebnis eine erhöhende Wirkung auf den Effekt, so ist die Funktion $abs(x)$ zu wählen.[239] Es ergibt sich der Ausdruck

$$f_j(u_j(n_a(j) - v_a(j))). \tag{5.14}$$

Darauf aufbauend können nun die direkten Effekte einer Aktion a über die Summe aller Dimensionen durch die Effektfunktion

$$e(a) = \sum_{j=1}^{n} w_j f_j(u_j(n_a(j) - v_a(j))) \tag{5.15}$$

[236] Beispiele: Energiestand in %, Anzahl Personen, Absolute Höhe in m, Station, Gegenwind in km/h, Temperatur in °C, ... usw.

[237] Beispiel: Der Wert des SOC (*state of charge*) der Dimension Energiestand in dem Zustand vor der Aktion „*fahre von Station A nach Station B*“ betrug 80% und im Zustand danach 60%. Die Differenz $80\% - 60\% = 20\%$ hat eine Erhöhung der Energiekosten von $id(u_j * 20\%) = u_j * 20\%$ zur Folge.

[238] Beispiel: Der Wert der Dimension Personenanzahl im Zustand vor der Aktion „*fahre mit aktiver Federung*“ betrug 5 Personen und im Zustand danach 8 Personen. Die Differenz $5P - 8P = -3P$ hat eine Erhöhung der Kompensationskosten von $-id(u_j * (-3P))) = u_j * 3P$ zur Folge.

[239] Beispiel: Der Wert der Dimension Absolute Höhe in den Zuständen bei der Fahrt durch ein Gebirge mit 2 Aktionen betrug $200m$, $500m$ und $100m$. Die positiven sowie negativen Veränderungen in der Höhe können in Energiekosten von $abs(u_j * (200m - 500m)) + abs(u_j * (500m - 100m)) = u_j * 700m =$ ausgedrückt werden.

quantifiziert werden. Ein Gewichtungswert w_j steuert zu welchem Maße der Wert einer jeweiligen Dimension j in die Gesamtbetrachtung mit einfließt.

Setzt man nun für Vor- und Nachbedingung beliebige Zustandsvektoren ein, so kann aus dem Effekt von Aktionen das Proximitätsmaß zur Quantifizierung der (Un)-Ähnlichkeit von Systemzuständen hergeleitet werden. Als Proximitätsmaß ergibt sich folgender Ausdruck:

$$d(\vec{s}_1, \vec{s}_2) = \sum_{j=1}^{n} w_j f_j(u_j(s_1(j) - s_2(j))) \tag{5.16}$$

mit $\vec{s}_1, \vec{s}_2 \in \mathbb{R}^n$. Zwei Zustände $\vec{s}_1$ und $\vec{s}_2$ haben hier die Proximität 0, wenn der Übergang von $\vec{s}_1$ nach $\vec{s}_2$ keinen direkten Effekt verursacht. Ähnliche Zustände weisen daher einen sehr geringen Effektabstand auf.

Algorithm 7: PROXIMITY

Input: $\vec{s}_1, \vec{s}_2, F_{1,...,n}, W_{1,...,n}, U_{1,...,n}$
Data: -
Output: proximity

```
1 proximity ⟵ 0;
2 foreach dim_j of s⃗_1, s⃗_2 do
3     proximity ⟵ proximity + W_j * F_j(U_j * (s⃗_2(j) − s⃗_1(j)));
4 return proximity;
```

Der Algorithmus 7 zeigt das Vorgehen zum Ermitteln der Proximität zweier Zustände s_1 und s_2 bei gegebenen Folgen von Funktionen F, Gewichtungswerten W und Normalisierungsfaktoren U.

5.1.2.3 Partitionierende Clusteranalyse von Zuständen

In diesem Abschnitt ist die Durchführung der partitionierenden Clusteranalyse zur Bildung der Zustandsklassen des (erweiterten) adaptiven MDPs (vgl. Abs. 5.1.2.1) Gegenstand näherer Betrachtung. Dabei wird das Clusteringverfahren *k-Means* aus der Literatur[240] entnommen und an das vorliegende Modell angepasst.

Gegeben seien die in einem (erweiterten) adaptiven MDP zu klassifizierende Zustände $\vec{s}_1, ..., \vec{s}_n$. Diese gilt es aufgrund ihrer Zustandseigenschaften in K Zustandsklassen $C_1, ..., C_K$ zu partitionieren, sodass jede Klasse über n_k Zustände verfügt und jeder Zustand genau einer Klasse zugeordnet ist. Demzufolge gilt

$$\sum_{k=1}^{K} n_k = n.^{241} \tag{5.17}$$

Als zu betrachtende Eigenschaft x eines Zustandes s wird der Zustandsvektor durch

$$\vec{x}_s = \vec{s} = (s(1), ..., s(d)) \tag{5.18}$$

[240] Vgl. [JD88], S. 89 ff.
[241] Vgl. [JD88], S. 93.

mit d für die Anzahl an Dimensionen und $\vec{s} \in \mathbb{R}^d$ gebildet. Ein Zustand s ist daher das Objekt, das einer Klasse C_k angesichts der Eigenschaft $\vec{x}_s$ zugeordnet wird. Der Schwerpunkt $\vec{m}_k$ bezeichnet den Repräsentanten einer Klasse C_k. Hier wird dieser Schwerpunkt durch

$$\vec{m}_k = (1/n_k) \sum_{i=1}^{n_k} \vec{x}_s \tag{5.19}$$

mit $\vec{x}_s \in C_k$ berechnet.[242] Die Zuordnung der Zustände mit ihren Eigenschaften erfolgt nach dem zuvor in Abschnitt 5.1.2.2 entwickelten Proximitätsmaß zwischen der Zustandseigenschaft $\vec{x}_s$ und dem Schwerpunkt $\vec{m}_k$ einer Klasse C_k. Ein Zustand s wird der Klasse zugeordnet, zu der die Proximität (vgl. Ausdruck 5.16) zum Schwerpunkt minimal ist:

$$\forall_{\vec{m}_1,...,\vec{m}_K} \min d(\vec{m}_k, \vec{x}_s). \tag{5.20}$$

Da zu garantieren ist, dass die Klassenschwerpunkte gültige Zustände repräsentieren, wird als tatsächlicher Schwerpunkt der Zustandsvektor mit der geringsten Proximität zum mathematischen Schwerpunkt festgelegt.

Basierend auf diesen Vorüberlegungen sind nun zur Durchführung einer partitionierenden Clusteranalyse folgende Schritte hintereinander auszuführen:[243]

1. Wählen von K initialen Klassenschwerpunkten
2. Wiederholen der Schritte 2 bis 5, solange bis sich die Klassenzugehörigkeit stabilisiert hat
3. Generieren einer neuen Partition durch Zuordnen aller Zustände $s_1, ..., s_n$ zu ihrem nächstliegenden Klassenschwerpunkt m_k
4. Berechnen neuer Klassenschwerpunkte $\vec{m}_1, ..., \vec{m}_K$
5. Wiederholen der Schritte 2 und 3 bis alle Proximitäten minimal sind
6. Anpassen der Klassengrößen durch Reduzieren von Zuständen

Zur Auswahl initialer Klassenschwerpunkte (vgl. Punkt 1) existieren in der Literatur verschiedene Strategien.[244] Bei diesen werden häufig, hinsichtlich der Proximität, weit auseinanderliegende Schwerpunkte bevorzugt. In dieser Arbeit steht die Verhaltensantizipation charakteristischer Ausführungsverläufe des kognitiven mechatronischen Systems im Vordergrund. Die Auswahl der initialen Schwerpunkte ist daher domänenspezifisch und in Abhängigkeit des vorliegenden Planungsproblems zu treffen.

[242] Vgl. [JD88], S. 93.
[243] In Anlehnung an [JD88], S. 96 f.
[244] Vgl. [JD88], S. 98.

Algorithm 8: DO-CLUSTERING

Input: $l_{cap}^{p'}$
Data: Σ^{Ext}
Output: -

```
1  hasChanged ⟵ true
   /* repetition until the clusters do not change any longer      */
2  while hasChanged do
3      hasChanged ⟵ false
4      minSum ⟵ +∞
5      sum ⟵ 0
       /* repetition until all proximities are minimal            */
6      while sum ≠ minSum do
7          minSum ⟵ sum
8          sum ⟵ 0
           /* generating a new partition                          */
9          foreach sn_i^C of Σ^Ext do
10             m⃗_new ⟵ (0, ..., 0);
11             n_k ⟵ sn_i^C.C.l^conditions.size
12             foreach s⃗_j of l_i^conditions do
13                 sum ⟵ sum + proximity(sn_i^C.C.m⃗, x⃗_{s_j})
14                 m⃗_new ⟵ m⃗_new + x⃗_{s_j}
15             m⃗_new ⟵ 1/n_k * m⃗_new
16             s⃗_min ⟵ nil
17             d_min ⟵ +∞
18             foreach s⃗_j of l_i^conditions do
19                 d ⟵ proximity(m⃗_new, x⃗_{s_j})
20                 if d < d_min then
21                     d_min ⟵ d
22                     s⃗_min ⟵ s⃗_j
23             if s⃗_min ≠ sn_i^C.C.m⃗ then
24                 hasChanged ⟵ true
25             sn_i^C.C.m⃗ ⟵ s⃗_min
26             sn_i^C.l^plans.clear()
27             sn_i^C.C.l^conditions.clear()
28         Σ^Ext.A.clear()
           /* rebuilding adaptive mdp with new state node clusters */
29         foreach p'_i of l_cap^{p'} do
30             classifyPlan(p'_i)
```

Darunter ist Folgendes zu verstehen: Geht man von der Beispieldomäne eines Transportsystems aus, so besteht ein Auftrag beispielsweise darin, Personen oder Güter von Station A nach Station B zu transportieren. Die Strecke zwischen diesen beiden Stationen beinhaltet mehrere Zwischenstationen. Bei der Planung zur Erfüllung des Auftrages erstellt das Transportsystem, z. B. zur Steuerung eines Federsystems bis zur nächsten Zwischenstation, im Voraus einen bedingten Plan mit alternativen Aktionen (bzw. Operationsmodi). Nach jedem Erreichen einer Zwischenstation wird der ausgeführte Plan mittels eines (erweiterten) adaptiven MDPs erfasst. Hat das Transportsystem einen Auftrag vollständig abgeschlossen, so wurde zu jedem der Zwischenstationen ein Zustand im (erweiterten) adaptiven MDP dokumentiert. Jeder weitere zu einer Zwischenstation dokumentierte Zustand stellt eine Wiederholung oder häufiger einen alternativen Zustand aufgrund von sich ändernden Gegebenheiten im Systemumfeld dar (z. B. Streckeneigenschaften, Umweltbedingungen oder Einfluss anderer Systeme). Diese alternativen Zustände gilt es mit Bezug zur jeweiligen Zwischenstation im Hinblick auf ihre Ähnlichkeit in Klassen zusammenzufassen. Infolgedessen werden in diesem Beispiel für jede der Stationen (inkl. Zwischenstationen) initiale Klassenschwerpunkte gewählt, die ein eindeutiges Merkmal zur Identifikation der Station enthalten und ein typisches Spektrum von Zuständen des Transportsystems in der Station abdecken.

Der letzte Schritt des Clusteringverfahrens (vgl. Punkt 5) beinhaltet in der Regel eine Anpassung der Anzahl an Klassen durch Zusammenführen und Teilen existierender Klassen oder Entfernen kleiner oder ausreißender Klassen[245], d. h. die Anzahl der Klassen wird verändert. Hier wird die Anzahl an Klassen, aus den zuvor beschriebenen Gründen, beibehalten und stattdessen die Größe der Klasse selbst z. B. durch das Reduzieren von Zuständen angepasst.

Der Algorithmus 8 zeigt den Ablauf einer partitionierenden Clusteranalyse auf dem (erweiterten) adaptiven MDP bei einer festen Anzahl von K Klassen. Mithilfe dieser Clusteranalyse können von den im Zeitverlauf erfassten Zuständen, entsprechend ihrer Eigenschaften, Klassen gebildet und alternative Ausführungsverläufe mit ähnlichen Ausgangs- und Zielzuständen gefunden werden.

5.1.3 Klassifizierung von Plänen

Die im vorherigen Abschnitt beschriebene Clusteranalyse (vgl. Abs. 5.1.2) dient ausschließlich zur Klassenbildung bereits erfasster Zustände. Mit der im Zeitverlauf steigenden Anzahl von Zuständen ist das Durchführen einer vollständigen Clusteranalyse in jedem Planungszyklus für einen neu erfassten Plan unter Echtzeitbedingungen des kognitiven mechatronischen Systems (vgl. hierzu auch Abb. 5.1) nicht mehr zielführend. Daher findet bei der Erfassung eines neuen Planes p'_i unter Echtzeitbedingungen mit dem (erweiterten) adaptiven MDP (vgl. Abs. 5.1.2.1) alternativ eine Klassifizierung statt. Die Aktualisierung der Klassenschwerpunkte bzw. die Rekonstruktion des (erweiterten) adaptiven MDPs durch Minimierung

[245] Vgl. z. B. [JD88], S. 98.

aller Distanzen ist dann ausschließlich in Ruhephasen des Systems oder beispielsweise in einem echt-parallel ausgeführten Prozess einer separaten Rechnerinstanz durchzuführen.

Zu diesem Zweck werden nun die bei der Erfassung von Plänen ursprünglich auf dem adaptiven MDP verwendeten Methoden zur Suche (vgl. Abs. 5.1.3.1), zum Hinzufügen (vgl. Abs. 5.1.3.2) und Entfernen (vgl. Abs. 5.1.3.3) von Plänen durch neue klassifikationsbasierte Methoden zur Klassenermittlung, Klassifizierung bzw. Deklassifizierung von Plänen ersetzt (vgl. Abs. 5.1.3.1 bis 5.1.3.3).

Wie ursprünglich, wird durch das Anwenden der neuen Methoden ein (erweiterter) adaptiver MDP im Zeitverlauf auf- bzw. abgebaut. Der Unterschied besteht lediglich darin, dass bei der Klassifizierung von Plänen davon ausgegangen wird, dass eine feste Anzahl an K Zustandsklassen mit entsprechenden Zuständen in dem vorliegenden (erweiterten) adaptiven MDP bereits existiert.

5.1.3.1 Klassifikationsbasiertes Suchen von Pänen

Die klassifikationsbasierte Suche nach einem zum Zustand $\vec{s}$ zugehörigen Zustandsklassenknoten erfolgt durch die Minimierung der Proximität zwischen den Schwerpunkten $\vec{m}_1, ..., \vec{m}_K$ und dem Eigenschaftsvektor $\vec{x}_s$. Im Vergleich zur ursprünglichen Methode zum Hinzufügen von Plänen (vgl. Abs. 5.1.1.2) wird jetzt also nicht mehr nach einem einzelnen Zustand gesucht, sondern nach einer Klasse mit ggf. mehreren Zuständen. Der Algorithmus 9 zeigt den Ablauf zur klassifikationsbasierten Suche nach der zum Zustand $\vec{s}$ zugehörigen Klasse sn^C_{min} bei gegebenem (erweiterten) adaptiven MDP Σ^{Ext}.

Algorithm 9: SEARCH-STATECLUSTERNODE

Input: $\vec{s}$
Data: Σ^{Ext}
Output: sn^C_{min}

```
1 d_min ⟵ +∞
2 sn^C_min ⟵ nil
3 foreach sn^C_s of Σ^Ext do
4     d ⟵ proximity(sn^C_s.C.m⃗, x⃗_s)
5     if d < d_min then
6         d_min ⟵ d
7         sn^C_min ⟵ sn^C_s
8 return sn^C_min
```

Mithilfe dieser neuen Suche kann das klassifikationsbasierte Hinzufügen und Entfernen von Plänen formalisiert werden.

5.1.3.2 Klassifikationsbasiertes Hinzufügen von Plänen

Das klassifikationsbasierte Hinzufügen eines neuen Planes p'_i zu einem (erweiterten) adaptiven MDP besteht nun in einer Klassifizierung des Planes. Genau wie in der Ursprungsmethode zum Hinzufügen von Plänen (vgl. Abs. 5.1.1.2) ist eine Fallunterscheidung zur Aktualisierung der Zählvariablen und zum Hinzufügen von Zuständen und Kanten gegeben. Der Algorithmus 10 zeigt den Ablauf zum Klassifizieren eines Planes p'_i.

Algorithm 10: CLASSIFY-PLAN

Input: p'_i
Data: Σ^{Ext}
Output: -

```
/* searching for state and plan nodes                                    */
```
1 $sn^C_{pre} \longleftarrow \Sigma^{Ext}$.`searchStateClusterNode`$(p'_i.\vec{v})$
2 $pn_s \longleftarrow \Sigma^{Ext}$.`searchPlanNode`$(p'_i.l^{actions})$
3 $sn^C_{post} \longleftarrow \Sigma^{Ext}$.`searchStateClusterNode`$(p'_i.\vec{n})$
```
/* updating state cluster and plan nodes                                 */
```
4 $sn^C_{pre}.C.l^{conditions}$.`add`$(p'_i.\vec{v})$
5 $sn^C_{post}.C.l^{conditions}$.`add`$(p'_i.\vec{n})$
6 $sn^C_{pre}.\#d \longleftarrow sn^C_{pre}.\#d + 1$
7 $sn^C_{post}.\#d \longleftarrow sn^C_{post}.\#d + 1$
8 **if** $pn_s = nil$ **then**
9 | $pn_s \longleftarrow (l_i^{actions}, 1, ())$
10 | Σ^{Ext}.`addPlanNode`(pn_s)
11 **else**
12 | $pn_s.\#a \longleftarrow pn_s.\#a + 1$
13 ...
```
/* ... for the next lines cf. algorithm 5                                */
```

Da weiterhin das Erfassen und Klassifizieren von Plänen in einem mitlaufenden und von Δ abhängigen Zeitfenster zu gewährleisten ist, muss ebenfalls eine korrespondierende Erweiterung zum Entfernen von Plänen erstellt werden.

5.1.3.3 Klassifikationsbasiertes Entfernen von Plänen

Das klassifikationsbasierte Entfernen eines im (erweiterten) adaptiven MDP erfassten Planes besteht nun in einer analogen Umsetzung seiner Deklassifizierung. Genau wie in der Ursprungsmethode zum Entfernen von Plänen (vgl. Abs. 5.1.1.3) ist eine Fallunterscheidung zur Aktualisierung der Zählvariablen und zum Entfernen von Zuständen und Kanten gegeben. Der Algorithmus 11 zeigt den zugehörigen Ablauf zum Deklassifizieren eines Planes p'_i.

In den vorangegangenen Abschnitten wurde gezeigt, wie das ursprüngliche Modell des adaptiven MDPs um Zustandsklassen erweitert werden kann und Verfahren

bzw. Methoden zur Klassenbildung sowie Klassifizierung umgesetzt werden können. Dies stellt den Umgang mit den (quasi-)kontinuierlichen Zustandswerten des kognitiven mechatronischen Systems im prädiktiven Modell der Top-Ebene sicher (vgl. hierzu auch Abs. 2.2.3.2).

Algorithm 11: DECLASSIFY-PLAN

Input: p'_i
Data: Σ^{Ext}
Output: -

```
/* searching for state and plan nodes                                    */
```
$sn^C_{pre} \longleftarrow \Sigma^{Ext}$.searchStateClusterNode$(p'_i.\vec{v})$
$pn_s \longleftarrow \Sigma^{Ext}$.searchPlanNode$(p'_i.l^{actions})$
$sn^C_{post} \longleftarrow \Sigma^{Ext}$.searchStateClusterNode$(p'_i.\vec{n})$
```
/* updating state cluster and plan nodes                                 */
```
if $sn^C_{pre}.\#d > 0$ **then**
    $sn^C_{pre}.C.l^{conditions}$.remove$(p'_i.\vec{v})$
    $sn^C_{pre}.\#d \longleftarrow sn^C_{pre}.\#d - 1$
if $sn^C_{post}.\#d > 0$ **then**
    $sn^C_{post}.C.l^{conditions}$.remove$(p'_i.\vec{n})$
    $sn^C_{post}.\#d \longleftarrow sn^C_{post}.\#d - 1$
if $pn_s.\#a = 1$ **then**
    Σ^{Ext}.removePlanNode(pn_s)
else
    $pn_s.\#a \longleftarrow pn_s.\#a - 1$
...
```
/* ... for the next lines cf. algorithm 6                                */
```

Auf Basis des (erweiterten) adaptiven MDPs und den entwickelten klassifikationsbasierten Methoden kann nun die Verhaltensantizipation und -regelung aufgebaut werden.

5.2 Verhaltensantizipation

Ziel dieses Teilkapitels ist das Entwickeln der Verhaltensantizipation für kognitive mechatronische Systeme. Unter Verwendung eines (erweiterten) adaptiven MDPs (vgl. Abs. 5.1.2.1) auf einer Top-Ebene werden innerhalb der hierarchischen Verhaltensplanung und -regelung (vgl. Abs. 2.2.1) die Auswirkungen von unterschiedlichen Entscheidungsalternativen der überlagerten Basis-Ebene vorweggenommen und entsprechend bewertet. Auswirkung meint hier die ausgehend vom aktuellen Systemzustand zu erwartenden direkten Kosten bei der Vorwegnahme einer Folgeaktion bzw. eines als Nächstes auszuführenden (Teil)-Planes. Um insb. das langfristige Verhalten, das über den begrenzten Planungshorizont der überlagerten Basis-

Ebene hinausgeht (vgl. Abs. 2.2.3.1), antizipieren zu können, wird an dieser Stelle eine hinreichende Anreicherung des (erweiterten) adaptiven MDPs durch bereits erfasste und klassifizierte Pläne vorausgesetzt (vgl. Abs. 5.1). Mittels des Echtzeit-Iterationsverfahrens RTDP (vgl. Abs. 3.2.1) wird im Folgenden, ausgehend vom aktuellen Systemzustand, der im Sinne der direkten Kosten und aus langfristiger Sicht günstigste (Teil-)Plan als Vorgabe zur Steuerung der Basis-Ebene bestimmt (vgl. Instruktion, in Abs. 2.2.1.1).

In jedem Zyklus der ereignisorientiert angestoßenen rollierenden Verhaltensplanung (vgl. Abs. 2.2.2) findet nun zusätzlich, und im Anschluss an die Erfassung und Klassifikation des zuletzt ausgeführten Teilplanes, eine Verhaltensantizipation statt (vgl. Gesamtablauf, Abb. 5.1). Zur Durchführung der Verhaltensantizipation werden folgende Schritte hintereinander durchlaufen:

1. Ermitteln des Zustandsklassenknotens sn_s^C im gegebenen (erweiterten) adaptiven MDP Σ^{Ext}, der die ähnlichste Situation zum aktuellen Systemzustand des kognitiven mechatronischen Systems repräsentiert (vgl. Abb. 5.10, sn_s^C)
2. Ermitteln der Menge an Zustandsklassenknoten SN_g^C im gegebenen (erweiterten) adaptiven MDP Σ^{Ext}, die gültige Zielzustände der Top-Ebene h_{j-1} repräsentieren (vgl. Abb. 5.10, SN_g^C)
3. Anwenden des Echtzeit-Iterationsverfahrens RTDP mit Startknoten sn_s^C und Zielknotenmenge SN_g^C zum Bestimmen des (Teil-)Planes π^* mit den minimal zu erwartenden direkten Kosten im Zustandsklassenknoten sn_s^C

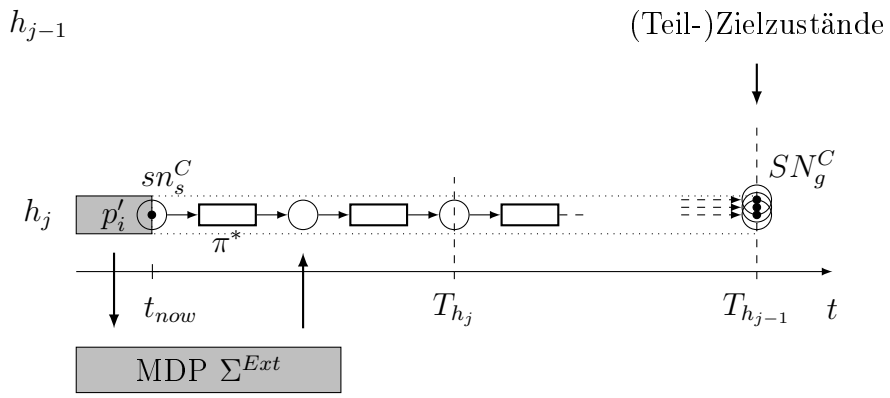


Abbildung 5.10: Planungshorizontübergreifende Verhaltensantizipation auf Basis eines (erweiterten) adaptiven MDPs

Weil die Möglichkeit zur Bewertung der Entscheidungsalternativen im Zusammenhang mit RTDP von zentraler Bedeutung ist, folgt im nächsten Abschnitt zunächst die Konzeption der Kostenbewertung zum Berechnen der direkten Kosten von (Teil-)Plänen und der zu erwartenden direkten Kosten in Zustandsklassen (vgl. Abs. 5.2.1). Im Anschluss daran findet die Konzeption der verhaltens- und ergebnisorientierten Echtzeit-Antizipation auf Basis von RTDP statt. Dabei sind insb. die Methoden zum Bestimmen der situations- und zielkonformen Zustandsklassen im (erweiterten) adaptiven MDP Gegenstand näherer Betrachtung (vgl. Abs. 5.2.2).

5.2.1 Kostenbewertung

Um die aus langfristiger Sicht kostenminimale Stratgie auf Basis eines gegebenen (erweiterten) MDPs bestimmen zu können, müssen zum einen die erfassten Pläne hinsichtlich ihrer direkten Kosten bewertet und zum anderen die zu erwartenden direkten Kosten in den Zustandsklassenknoten berechnet werden.

Im Abschnitt 5.1.2.2 wurde bereits die Proximität von Systemzuständen des kognitiven mechatronischen Systems aus den direkten Effekten von Aktionen hergeleitet und dabei die unterschiedlichen Dimensionen der Zustände mitberücksichtigt. Weil die in einem Planknoten des (erweiterten) MDPs erfasste Aktionsfolge (Plan) eine Überführung von erster Vorbedingung zu letzter Nachbedingung repräsentiert, kann die bereits entwickelte Effektfunktion (vgl. Ausdruck 5.15) wieder aufgegriffen und in angepasster Form zur Kostenbewertung einer ganzen Aktionsfolge bzw. eines Planes verwendet werden. Aus diesem Grund sind die direkten Kosten eines Planes p' durch die folgende Kostenfunktion bestimmbar:

$$c(\vec{v}, \vec{n}) = \sum\nolimits_{j=1}^{n} w_j f_j(u_j(v(j) - n(j))) \tag{5.21}$$

mit $\vec{v}, \vec{n} \in \mathbb{R}^n$ und $\vec{v}$ als Vorbedingung und $\vec{n}$ als Nachbedingung von p'. An dieser Stelle sei darauf hingewiesen, dass die aus dem (erweiterten) adaptiven MDP zur Kostenbewertung eines Planes eingesetzten Vor- und Nachbedingungen den Klassenschwerpunkten der beteiligten Zustandsklassen entsprechen (vgl. Abs. 5.1.2.3) und damit gültige Zustände des kognitiven mechatronischen Systems repräsentieren. Mithilfe dieser Kostenfunktion können verschiedene (Teil-)Pläne miteinander verglichen und für die aktuelle Situation eine kostenminimale Auswahl getroffen werden.

Die zuvor beschriebene Kostenfunktion ist allerdings nur ausreichend, wenn man davon ausgeht, dass ausgehend vom aktuellen Systemzustand (Vorbedingung) die Ausführung des gewählten Planes auch mit absoluter Sicherheit in den, während der Kostenbewertung miteinbezogenen, Folgezustand (Nachbedingung) führt. Da dies aber aufgrund der nicht-deterministischen Systemumgebung nicht garantiert ist, sind zum Bewerten der Pläne und zur Quantifizierung der zu erwartenden direkten Kosten in Zustandsklassenknoten zusätzlich die Gewichtungen der ausgehenden Kanten des jeweils gegebenen Planknotens innerhalb des (erweiterten) adaptiven MDP miteinzubeziehen. Es müssen also alle möglichen Folgezustände des gewählten Planes in die Kostenbewertung einfließen (vgl. Abs. 5.1.1).

In den nächsten beiden Abschnitten wird gezeigt, wie das Bewerten direkter Kosten von Plänen (vgl. Abs. 5.2.1.1) und darauf aufbauend das Bewerten der zu erwartenden direkten Kosten in Zustandsklassen (vgl. Abs. 5.2.1.2) auf Basis des (erweiterten) adaptiven MDPs durchgeführt werden können.

5.2.1.1 Direkte Kosten von Plänen

Die Abbildung 5.11 zeigt den allgemeinen Ausschnitt eines (erweiterten) adaptiven MDPs (vgl. Abs. 5.1.2.1) mit drei Zustandsklassenknoten (inkl. Klassenschwerpunk-

te, vgl. Abs. 5.1.2.3) und einem Planknoten mit zugehöriger Kantengewichtung. Das Ziel besteht nun darin, ausgehend von einem Zustandsklassenknoten sn_k^C, die direkten Kosten bei zunächst fixiertem Planknoten pn_i zu berechnen.

Die Schwerpunkte der Zustandsklassen (vgl. Abb. 5.11, $\vec{m}_k$ und $\vec{m}_j$) repräsentieren Vor- und Nachbedingungen (bzw. Systemzustände), die zu allen anderen in der Klasse erfassten Zuständen am ähnlichsten sind. Zur Erinnerung: Als Repräsentant einer Klasse wurde der Zustand mit der minimalen Proximität zum tatsächlichen geometrischen Klassenschwerpunkt gewählt. Damit ist garantiert, dass es sich bei diesem auch um einen gültigen Systemzustand handelt (vgl. Abs. 5.1.2.3). Auf Basis dieser Repräsentanten erfolgt hier die Berechnung der direkten Kosten bzgl. der untersuchten Planausführung. Genauer betrachtet: Der durchschnittlichen direkten Kosten.[246]

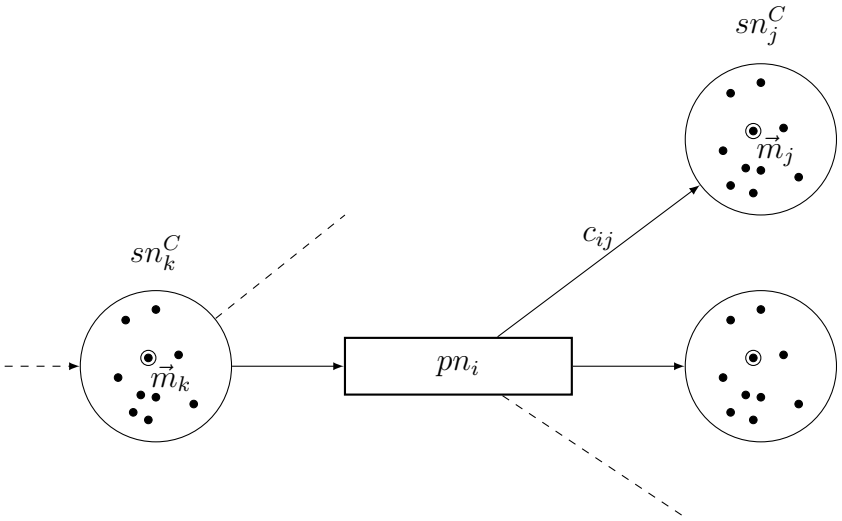


Abbildung 5.11: Ausschnitt eines (erweiterten) adaptiven MDPs

Zu diesem Zweck werden ausgehend von der Vorbedingung $\vec{m}_k$ des beteiligten Planknotens pn_i zu allen erreichbaren Nachbedingungen $\vec{m}_j$ die direkten Kosten gemäß des Ausdrucks 5.21 berechnet. Mithilfe der gewichteten Summe unter Einbezug der gegebenen Kantengewichte c_{ij} können so die durchschnittlichen direkten Kosten bestimmt werden. Zur Berechnung der durchschnittlichen direkten Kosten zu einem Planknoten pn_i, ausgehend von einem Zustandsklassenknoten sn_k^C, ergibt sich daher folgender Ausdruck:

$$c_{avg}(sn_k^C, pn_i) = \sum_{j=1}^{n} c_{ij} * c(\vec{m}_k, \vec{m}_j) \tag{5.22}$$

mit $\vec{m}_k$ als Klassenschwerpunkt von sn_k^C und $\vec{m}_j$ als Klassenschwerpunkt von sn_j^C, wobei sn_k^C Vorbedingungen, sn_j^C Nachbedingungen und n die Anzahl der ausgehenden Kanten von Planknoten pn_i sind.

[246] Mit einer Auswertung über das gesamte Spektrum der in einem Zustandsklassenknoten vorkommenden Zustände wäre auch eine Bewertung der maximalen oder minimalen direkten Kosten denkbar.

Der Algorithmus 12 zeigt den Ablauf zum Berechnen der durchschnittlichen direkten Kosten eines im Zustandsklassenknoten sn_i^C ausführbaren Planknotens pn_i. Weil die Berechnungsgrundlage nach wie vor durch die Proximität gebildet wird, und diese abhängig von den Funktionen F, Gewichtungswerten W und Normalisierungsfaktoren U ist (vgl. Abs. 5.1.2.2), müssen zur tatsächlichen Berechnung der Kostenwerte (z. B. in Euro) diese Parameter für die beteiligten Dimensionen entsprechend definiert werden.

Algorithm 12: VALUE-PLAN

Input: $sn_k^C, pn_i, F_{1,\ldots,n}, W_{1,\ldots,n}, U_{1,\ldots,n}$
Data: Σ^{Ext}
Output: c_{avg}

$c_{avg} \longleftarrow 0$
foreach c_{ij}^{state} **of** $pn_i.l^{states}$ **do**
  $sn_j^C \longleftarrow c_{ij}^{state}.sn^C$
  $c \longleftarrow c_{ij}^{state}.c$
  $c_{avg} \longleftarrow c_{avg} + c * \texttt{proximity}(sn_k^C.\vec{m}, sn_j^C.\vec{m}, F_{1,\ldots,n}, W_{1,\ldots,n}, U_{1,\ldots,n})$
$n \longleftarrow pn_i.l^{states}.\texttt{size}()$
return c_{avg}

Auf Grundlage dieser Vorüberlegungen können nun im nächsten Abschnitt die zu erwartenden direkten Kosten in einem Zustandsklassenknoten bestimmt werden.

5.2.1.2 Erwartete direkte Kosten in Zustandsklassen

Die in einem Zustandsklassenknoten sn_s^C zu erwartenden direkten Kosten sind abhängig von den dort aus zugänglichen Planknoten und dadurch erreichbare Folgezustandsklassenknoten. Zu jedem von sn_s^C aus erreichbaren Planknoten werden die direkten Kosten (vgl. Abs. 5.2.1.1) berechnet und dabei jeweils der aktuelle Kostenwert eines Folgezustandsklassenknotens mit einkalkuliert. Der angepasste Ausdruck zum Bestimmen der erwarteten Kosten eines in sn_s^C ausführbaren Planknotens pn_i ist hier gegeben durch

$$Q(sn_s^C, pn_i) = \sum_{j=1}^{n} c_{ij} * [c(\vec{m}_s, \vec{m}_j) + V(sn_j^C)] \tag{5.23}$$

mit $\vec{m}_s$ als Klassenschwerpunkt von sn_s^C und $\vec{m}_j$ als Klassenschwerpunkt von sn_j^C, wobei n die Anzahl der ausgehenden Kanten von Planknotens pn_i ist. Die zu erwartenden direkten Kosten in einem Zustandsklassenknoten können dann im Anschluss auf einfache Weise durch das Minimum der Planbewertungen über alle von sn_s^C aus erreichbaren Planknoten pn_i mittels

$$V(sn_s^C) = \min Q(sn_s^C, pn_i) \tag{5.24}$$

quantifiziert werden. Die zum Zustandsklassenknoten zugehörige Strategie π entspricht dann dem Planknoten mit den minimalen zu erwartenden direkten Kosten.

Der als letztes genannte Ausdruck entspricht dem sogenannten Bellman-Ausdruck (vgl. Abs. 3.2.2). Dieser dient auch hier als Grundlage zum iterativen Bestimmen der in einem Zustandsklassenknoten zu erwartenden minimalen direkten Kosten $V^*(sn_s^C)$ sowie insb. zum Finden der optimalen Strategie $\pi^*(sn_s^C)$. Genau auf dieser Basis können Lösungsverfahren wie z. B. *Policy Iteration*, *Value Iteration* oder *Real-Time Value Iteration* auf dem gegebenen adaptiven (erweiterten) MDP zum Bestimmen des kostenminimalen Planes angewendet werden (vgl. Abs. 3.2.2). Damit dieses Vorhaben gelingt, ist die Modellierung der Zustandsklassenknoten des (erweiterten) adaptiven MDPs abermals in adäquater Weise zu erweitern. Ein Zustandsklassenknoten wird nun modelliert durch das 5-Tupel

$$SN^C = (C, \#d, l^{plans}, v, pn_\pi) \tag{5.25}$$

mit zusätzlichem Planknoten (Strategie) pn_π und zusätzlicher Kostenvariablen v.

Unter Verwendung der gezeigten Kostenbewertung kann nun im nächsten Abschnitt die eigentliche Durchführung der RTDP-gestützten Verhaltensantizipation für kognitive mechatronische System entwickelt werden.

5.2.2 Verhaltens- und ergebnisorientierte Echtzeit-Antizipation

Ziel dieses Abschnittes ist die Konzeption der verhaltens- und ergebnisorientierten Echtzeit-Antizipation für kognitive mechatronische Systeme auf Basis von RTDP (vgl. Abs. 3.2.2). Ausgehend von einem gewählten Startzustandsklassenknoten sn_s^C wird im gegebenen (erweiterten) adaptiven MDP Σ^{Ext} eine bestimmte Anzahl von Pfaden zufällig exploriert. An jedem auf diesen Pfaden besuchten Zustandsklassenknoten findet eine Aktualisierung der Kostenwerte (Bellman-Update, vgl. Ausdruck 5.24) und damit ein sukzessives Bestimmen der kostenminimalen Pläne statt. Die Exploration eines Pfades wird dabei mit dem Erreichen eines Zustandsklassenknotens aus einer vorher festgelegten Menge SN_g^C von Zielzustandsklassenknoten terminiert. Für einen einzelnen Pfad bzw. eine einzelne Iteration sind daher in angepasster Form folgende Schritte durchzuführen:[247]

1. Berechnen der zu erwartenden direkten Kosten für jeden von Zustandsklassenknoten sn_s^C aus erreichbaren Planknoten pn (vgl. Ausdruck 5.23)
2. Bestimmen des Planknotens pn_{min} mit den minimalen zu erwartenden direkten Kosten (vgl. Ausdruck 5.24)
3. Aktualisieren des Kostenwerts im Zustandsklassenknoten sn_s^C mit den Kosten von pn_{min} (vgl. Ausdruck 5.24)
4. Zufällige Auswahl des Folgezustandsklassenknotens sn_j^C mit Wahrscheinlichkeit $P_{pn_{min}}(sn_j^C|sn_s^C) = c_{minj}$
5. Falls $sn_j^C \in SN_g^C$, beende den Pfad, sonst setze $sn_s^C = sn_j^C$ und gehe zu 1

[247] In Anlehnung an [BG01], [BH00] und [SGDS09].

Zur Umsetzung dieser Schritte innerhalb der Iteration einer RTDP-gestützten Echtzeit-Antizipation sind weitere vorbereitende Maßnahmen erforderlich. Zunächst muss ein Startzustandsklassenknoten im (erweiterten) adaptiven MDP festgelegt werden, zumal dieser den Ausgangspunkt zur Pfadexploration vorgibt. Daher wird im Folgenden zuerst das Bestimmen von situationskonformen Zustandsklassen behandelt (vgl. Abs. 5.2.2.1). Im Anschluss ist das Bestimmen von zielkonformen Zustandsklassen Gegenstand näherer Betrachtung (vgl. Abs. 5.2.2.2), weil die Pfadexploration wie gefordert durch eine Menge von Zielzustandsklassenkonten zu begrenzen ist. Da in jedem Explorationsschritt der Folgezustandsklassenkonten zufällig gewählt wird, findet weiterhin das Entwickeln einer randomisierten Selektion von Folgezustandsklassen statt (vgl. Abs. 5.2.2.3). Abschließend wird dann die eigentliche Durchführung des Gesamtverfahrens zum Ermitteln von kostenminimalen Instruktionen im Zusammenschluss konzipiert (vgl. Abs. 5.2.2.4).

5.2.2.1 Bestimmen von situationskonformen Zustandsklassen

Die aktuelle Situation, in der sich das kognitive mechatronische System befindet, wird in dieser Arbeit durch den aktuellen Zustandsvektor $\vec{s}$ des Systems repräsentiert. Zum Bestimmen eines situationskonformen Zustandsklassenknotens erfolgt eine Suche im gegebenen adaptiven (erweiterten) MDP Σ^{Ext} (vgl. Abs. 5.1.2.1) nach einem Zustandsklassenknoten sn_s^C, der Zustände kapselt, die möglichst ähnlich zum aktuellen Systemzustand sind (vgl. Abb. 5.12).

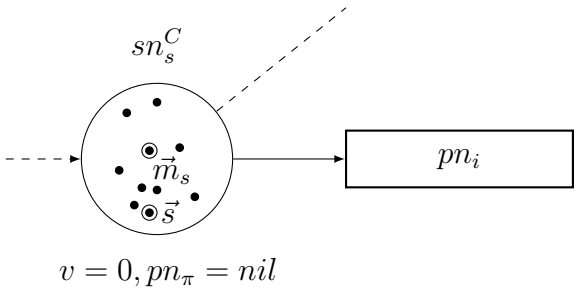


Abbildung 5.12: Situationskonformer Zustandsklassenknoten eines (erweiterten) adaptiven MDPs

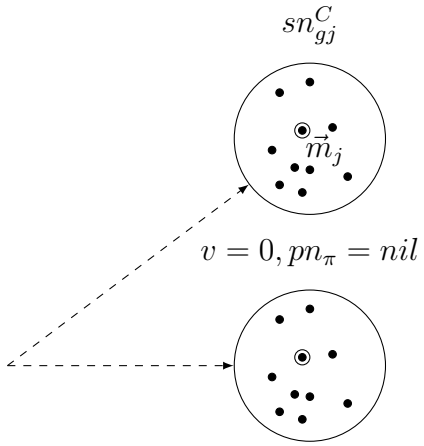
Weil der Klassenschwerpunkt $\vec{m}_s$ (vgl. Abb. 5.12) eines potentiellen Zustandsklassenknotens die in ihm enthaltenen Zustände repräsentiert, besteht die Suche vom Prinzip her in einer Klassifikation des aktuellen Systemzustands $\vec{s}$. Es wird daher im (erweiterten) adaptiven MDP Σ^{Ext} der Zustandsklassenknoten mit der minimalen Proximität (vgl. Abs. 5.1.2.2) zwischen Klassenschwerpunkt und dem aktuellen Systemzustand $\vec{s}$ gesucht. Folglich kann für diese Suche auf den bereits in Abschnitt 5.1.3.1 zur klassifikationsbasierten Suche von Zuständen entwickelten Algorithmus 9 zurückgegriffen werden.

Die minimalen zu erwartenden direkten Kosten sowie der optimale Plan sind zu diesem Zeitpunkt im Zustandsklassenknoten sn_s^C noch nicht bestimmt (vgl. Abb.

5.12).

5.2.2.2 Bestimmen von zielkonformen Zustandsklassen

Von einer überlagernden Planungsebene (vgl. Gesamthierarchie in Abs. 2.2.1.1) vorgegebene und durch das kognitive mechatronische System auf der überlagerten Planungsebene zu erfüllende Zielzustände bestehen in dieser Arbeit aus einer gegebenen Menge von Zustandsvektoren zukünftig anzustrebender Systemzustände $\vec{s}_{g1}, ..., \vec{s}_{gm}$. Zum Bestimmen von zielkonformen Zustandsklassenknoten erfolgt daher eine Suche im gegebenen adaptiven (erweiterten) MDP Σ^{Ext} (vgl. Abs. 5.1.2.3) nach einer Menge von Zustandsklassenknoten SN_g^C, die Zustände kapseln die diese Systemzustände bestmöglich erfüllen (vgl. Abb. 5.13). Weil auch in diesem Fall die Klassenschwerpunkte $\vec{m}_{g1}, ..., \vec{m}_{gm}$ potentieller Zustandsklassenknoten darin enthaltene Zustände repräsentieren, kann eine Suche nach zielkonformen Zustandsklassenknoten ebenfalls auf dieser Grundlage durchgeführt werden.



$$res_1 := \{x_{lower}(1) \leq m_j(1) \leq x_{upper}(1)\}$$
$$\vdots$$
$$res_i := \{x_{lower}(i) \leq m_j(i) \leq x_{upper}(i)\}$$
$$\vdots$$
$$res_n := \{x_{lower}(n) \leq m_j(n) \leq x_{upper}(n)\}$$

Abbildung 5.13: Zielkonforme Zustandsklassenknoten eines (erweiterten) adaptiven MDPs

Zu diesem Zweck ist die Definition einer Menge von erfüllbaren Zielzuständen durch das Formulieren von Restriktionen erforderlich, die für jede Dimension eines untersuchten Klassenschwerpunktes den erlaubten Wertebereich vorgibt. Für die Menge an zielkonformen Zustandsklassenknoten ergibt sich folgender Ausdruck:

$$\begin{aligned} SN_g^C := \{sn_j^C | \{x_{lower}(1) \leq m_j(1) \leq x_{upper}(1)\} \wedge \ldots \\ \cdots \wedge \{x_{lower}(i) \leq m_j(i) \leq x_{upper}(i)\} \wedge \ldots \\ \cdots \wedge \{x_{lower}(n) \leq m_j(n) \leq x_{upper}(n)\}\} \end{aligned} \tag{5.26}$$

mit $m_j(i) \in \vec{m}_j$, $x_{lower}(i), x_{upper}(i) \in \mathbb{R}$ und n für die Anzahl an Dimensionen von $\vec{m}_j$. Da es sich bei den zielkonformen Zustandsklassenknoten um Ziele bzw. Endpunkte der später untersuchten Pfade handelt, werden die Kostenwerte auf den Wert 0 gesetzt und die optimalen Pläne bleiben in diesem Knoten während des gesamten Verfahrens undefiniert (vgl. Abb. 5.13).

Der Algorithmus 13 zeigt die Suche nach einer Menge zielkonformer Zustandsklassenknoten SN_g^C in einem (erweiterten) adaptiven MDP Σ^{Ext} bei gegebenen Restriktionen $Res_{1...n}$.

Algorithm 13: SEARCH-GOAL-STATECLUSTERNODES

Input: $Res_{1...n}$
Data: Σ^{Ext}
Output: SN_g^C

```
1  SN_g^C ⟵ ()
2  foreach sn_s^C of Σ^Ext do
3      m⃗ ⟵ sn_s^C.m⃗
4      check ⟵ true
5      foreach res_i of Res_{1...n} do
6          if m⃗(i) < res_i.x_lower or m⃗(i) > res_i.x_upper then
7              check ⟵ false
8              break
9      if check then
10         SN_g^C.add(sn_s^C)
11 return SN_g^C
```

5.2.2.3 Randomisierte Selektion von Folgezustandsklassen

Das Anwenden des RTDP-Verfahrens auf einem (erweiterten) adaptiven MDP Σ^{Ext} umfasst u. a. die sukzessive Exploration von Zustandsklassenknoten. Ausgehend von einem Startzustandsklassenknoten sn_s^C (vgl. Abs. 5.2.2.1) werden einzelne Pfade bis zum Erreichen eines Zielzustandsklassenknotens (vgl. Abs. 5.2.2.2) Schritt für Schritt generiert. Die Wahl des nächsten Folgezustandsklassenknotens erfolgt dabei randomisiert und wird durch die Gewichtungen der ausgehenden Kanten des aktuell festgelegten Planknotens pn_i definiert. Diese bilden die Wahrscheinlichkeitsverteilung der zufälligen Selektion des Folgezustandsklassenkontens mit

$$P_{pn_i}(sn_j^C | sn_s^C) = c_{ij}. \tag{5.27}$$

Um den Suchraum gezielt einschränken zu können (z. B. bei einem Transportsystem das Betrachten von Folgezustandsklassenknoten mit Stationen in Fahrtrichtung einer bestimmten Route), wird hier die zusätzliche Boolesche Variable $isValid$ eingeführt. Diese soll sicherstellen, dass bei der randomisierten Selektion von Zustandsklassenknoten nur als valide gekennzeichnete Knoten gewählt werden. Wenn gezielt Zustandsklassenknoten ausgeschlossen werden, dann darf die Wahrscheinlichkeitsverteilung zur zufälligen Selektion nur durch die Gewichte von Kanten zu

validen Zustandsklassenknoten gebildet werden. Es folgt der Ausdruck

$$P_{pn_i}(sn_j^C|sn_s^C) = \frac{c_{ij}}{\sum_{m=1}^{|SN_{isValid}^C|} c_{im}} \tag{5.28}$$

mit $sn_j^C \in SN_{isValid}^C$.

Algorithm 14: RND-NEXT-STATE

Input: pn_i
Data: Σ^{Ext}
Output: sn_{next}^C

```
1  if pn_i.l^states.size() = 0 then
2      return nil
3  total ⟵ 0
4  foreach e_ij^state of pn_i.l^states do
5      sn_next^C ⟵ e_ij^state.sn
6      if not sn_next^C.isValid() then
7          total ⟵ total + e_ij^state.c
8  if total = 0 then
9      return nil
10 rnd ⟵ RNDFloat()
11 sum ⟵ 0
12 foreach e_ij^state of pn_i.l^states do
13     sn_next^C ⟵ e_ij^state.sn
14     if not sn_next^C.isValid() then
15         sum ⟵ sum + e_ij^state.c / total
16         if rnd < sum then
17             return sn_next^C
18 return nil
```

Der Algorithmus 14 zeigt den Ablauf zur randomisierten Selektion eines Folgezustandsklassenknotens bei gegebenem Planknoten pn_i.

Mithilfe der zuvor konzipierten Bausteine zum Bestimmen von situations- und zielkonformen Zustandsklassen sowie der randomisierten Selektion von Folgezustandsklassen kann nun die eigentliche RTDP-gestützte Echtzeit-Antizipation auf Basis des (erweiterten) adaptiven MDPs zum Bestimmen der kostenminimalen Instruktionen entwickelt werden.

5.2.2.4 Ermitteln von kostenminimalen Instruktionen

Dieser Abschnitt befasst sich mit der Konzeption zur vollständigen Durchführung der verhaltens- und ergebnisorientierten Echtzeit-Antizipation für kognitive me-

chatronische Systeme zum Ermitteln kostenminimaler Instruktionen. Ziel ist es, für einen situationskonformen Zustandsklassenknoten sn_s^C (vgl. Abs. 5.2.2.1), aus dem (erweiterten) adaptiven MDP auf der Top-Ebene, den aus langfristiger Sicht möglichst kostenminimalen Plan π^* zur Einplanung auf der Basis-Ebene zu ermitteln. Dabei wird zum einen auf die in den vorherigen Abschnitten entwickelten Methoden und zum anderen auf das bereits in der Literatur existierende Echtzeit-Iterationsverfahren RTDP (vgl. Abs. 3.2.2) zum Lösen von MDPs zurückgegriffen. Die verhaltens- und ergebnisorientierte Echtzeit-Antizipation liefert damit die entscheidende Grundlage für die später in dieser Arbeit zu integrierende Verhaltensregelung für kognitive mechatronische Systeme.

Neben einer zusätzlich eingeführten Planreichweite T_{max}, die verhindern soll, dass Pfade im (erweiterten) adaptiven MDP untersucht werden, die aufgrund ihrer Länge keine sinnvolle Lösung mehr darstellen (diese sollte mindestens der Planreichweite der überlagernden Planungsebene entsprechen, vgl. Abs. 2.2.3.1), wird außerdem die Menge der validen Zustandsklassenknoten $SN_{isValid}^C$ vorgegeben. Letzteres soll eine Einschränkung des Suchraumes auf die wesentlichen Zustandsklassenknoten vornehmen (z. B. bei einem Transportsystem ausschließlich Zustände mit Stationen, die auf der zu befahrenden Route liegen). Mithilfe der vorbereitenden Maßnahmen zum Bestimmen von situations- und zielkonformen Zustandsklassen (vgl. Abs. 5.2.2.1 und 5.2.2.2) sowie der randomisierten Selektion von Folgezustandsklassen (vgl. Abs. 5.2.2.3) kann nun die vollständige RTDP-gestützte Verhaltensantizipation für kognitive mechatronische Systeme umgesetzt werden. Zur Durchführung sind folgende Parameter zu wählen bzw. im Vorfeld zu bestimmen:[248]

1. Ein situationskonformer Startzustandsklassenknoten sn_s^C
2. Eine Menge SN_g^C von zielkonformen Zustandsklassenknoten
3. Die maximale Planreichweite T_{max}
4. Die Menge $SN_{isValid}^C$ von zulässigen Zustandsklassenknoten
5. Die Anzahl von Pfaden (*trails*)

Weil die Exploration einzelner Pfade hier nicht ausschließlich durch das Finden eines Zielzustandsklassenknotens, sondern zusätzlich mit dem Erreichen der Explorationstiefe bzw. Planreichweite T_{max} beendet wird, handelt es sich hierbei um eine begrenzte (engl. *limited*) (L)RTDP-gestützte Verhaltensantizipation. Falls kein vollständiger Pfad von Startzustandsklassenknoten zu einem Zielzustandsklassenknoten gefunden wird, garantiert dieses Vorgehen die Terminierung des Verfahrens.

Der Algorithmus 15 zeigt den Ablauf zur begrenzten (L)RTDP-gestützten Verhaltensantizipation bei gegebenem Startzustandsklassenknoten sn_s^C, einer Menge SN_g^C von Zielzustandsklassenknoten, einer maximalen Planreichweite T_{max}, einer Menge $SN_{isValid}^C$ von validen Zustandsklassenknoten sowie der Anzahl von zu untersuchenden Pfaden.

Die Konzeption der Verhaltensantizipation für kognitive mechatronische Systeme auf Basis eines (erweiterten) adaptiven Markov-Entscheidungsprozesses ist damit abgeschlossen und es kann nun darauf aufbauend die Integration der Verhaltensre-

[248] In Anlehnung an [BG01], [BH00] und [SGDS09].

gelung in die hierarchische Verhaltensplanung stattfinden.

Algorithm 15: ANTICIPATE-LRTDP

Input: $sn_s^C, SN_g^C, T_{max}, SN_{states}^C, trails$
Data: Σ^{Ext}
Output: -

foreach sn_i^C *of* SN_{states}^C **do**
  $sn_i^C.\pi \longleftarrow nil$
  $sn_i^C.V \longleftarrow 0$
$trail \longleftarrow 0$
while $trail < trails$ **do**
  $sn_{next}^C \longleftarrow sn_s^C$
  $d \longleftarrow 0$
  while $(sn_{next}^C \notin SN_g^C)$ ***and*** $(d < T_{max})$ ***and*** $(sn_{next}^C \neq nil)$ ***and*** $(sn_{next}^C.l^{plans}.size > 0)$ **do**
    $Q_{min_V} \longleftarrow +\infty$
    $Q_{min_{pn}} \longleftarrow nil$
    foreach e_{ij}^{plan} ***of*** $sn_{next}^C.l^{plans}$ **do**
      $plan \longleftarrow e_{ij}^{plan}.pn$
      $Q_v \longleftarrow 0$
      foreach e_{rs}^{state} ***of*** $plan.l^{states}$ **do**
        $state \longleftarrow e_{rs}^{state}.sn$
        $Q_v \longleftarrow Q_v + e_{rs}^{state}.c * (\texttt{valuePlan}(sn_{next}^C, plan) + state.V)$
      if $Q_v < Q_{min_V}$ **then**
        $Q_{min_V} \longleftarrow Q_v$
        $Q_{min_{pn}} \longleftarrow plan$
    $sn_{next}^C.V \longleftarrow Q_{min_V}$
    $sn_{next}^C.\pi \longleftarrow Q_{min_{pn}}$
    $sn_{next}^C \longleftarrow \texttt{rndNextState}(Q_{min_{pn}})$
    $d \longleftarrow d + 1$
  $trail \longleftarrow trail + 1$

5.3 Integration von Verhaltensregelung in die hierarchische Verhaltensplanung

Ziel dieses Teilkapitels ist die abschließende Integration einer MDP-basierten antizipatorischen Verhaltensregelung in die hierarchische Verhaltensplanung für kognitive mechatronische Systeme (vgl. Abs. 2.2.1.1). Es soll fortwährend die Verhaltensplanung (vgl. hybride Planungsarchitektur in Abs. 3.1.1) einer zeitlich kürzeren und inhaltlich detaillierenden Basis-Ebene von der ihr direkt übergeordneten zeitlich längeren und inhaltlich vergröbernden Top-Ebene (vgl. Abs. 2.2.1) im Sinne einer Angleichung an das übergeordnete Ziel der Kostenminimierung beeinflusst und so der langfristig ökonomische sowie autonome Betrieb des kognitiven mechatronischen Systems realisiert werden. Die Abbildung 5.14 zeigt die Wirkzusammenhänge zu zwei für die Gesamthierarchie repräsentativen Planungsebenen der hier angestrebten MDP-basierten antizipatorischen Verhaltensregelung.

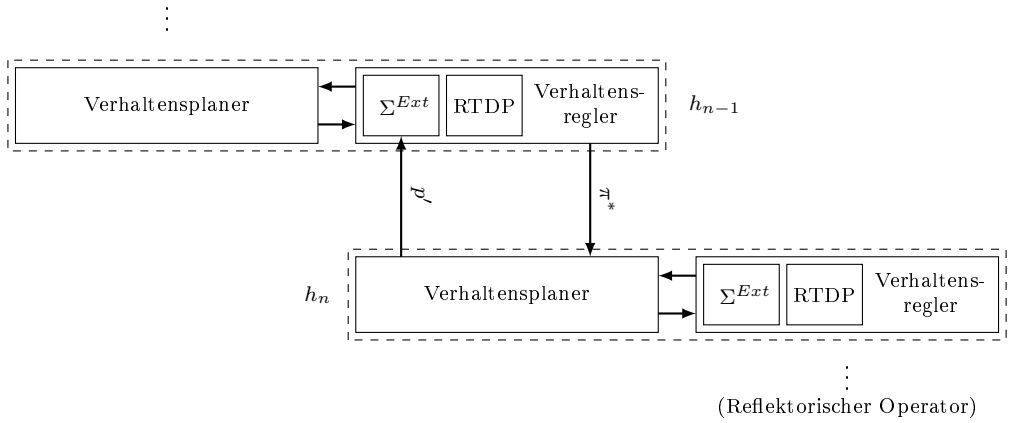


Abbildung 5.14: Wirkzusammenhänge einer MDP-basierten antizipatorischen Verhaltensregelung

Auf Basis des (erweiterten) adaptiven MDPs (vgl. Abs. 5.1.2.1) der regelnden Hierarchieebene (vgl. Abb. 5.14, Σ^{Ext}) wird das zukünftige Verhalten der geregelten Planungsebene antizipiert und die zugehörige Verhaltensplanung (vgl. Abs. 2.1.3, die Regelstrecke) durch Vorgabe einzuplanender (Teil-)Pläne (die Stellgröße) gesteuert. Zur zielkonformen Steuerung auf der regelnden Ebene bestimmt der Verhaltensregler mithilfe der Verhaltensantizipation (vgl. Abs. 5.2) die möglichst kostenminimale, und aus Sicht der detaillierenden Planungsebene, längerfristige Stellstrategie (vgl. Abb. 5.14, π^*). Zu diesem Zweck findet ausgehend vom aktuellen Systemzustand eine RTDP-gestützte Verhaltensantizipation (vgl. Abs. 5.2.2) in Richtung der Ziele (die Führungsgröße) statt, die durch die eingestellten und zu realisierenden Teilpläne der wiederum übergeordneten Ebene vorgegebenen sind. Mit dem Erfassen und Klassifizieren der tatsächlich ausgeführten Pläne, inkl. Vor- und Nachbedingungen (vgl. Abs. 5.1 und Abb. 5.14 p', die Rückführung), und der damit einhergehenden Aktualisierung des (erweiterten) adaptiven MDPs schließt

sich der Regelkreis der jeweils betrachteten regelnden Ebene. Mit der Kostenbewertung der Pläne ergibt sich dann indirekt die quantifizierte und zu regelnde Größe (die Regelgröße) in Form der zu erwartenden direkten Kosten.

In jedem Zyklus eines Regelkreises zwischen zwei Planungsebenen werden bei der Verhaltensregelung folgende Schritte wiederholt durchlaufen:

1. Durchführen einer RTDP-gestützten Antizipation mit dem (erweiterten) adaptiven MDP Σ^{Ext} zum Bestimmen des kostenminimalen Planes π^* für die geregelte Ebene in Abhängigkeit des eingeplanten Planes der regelnden Ebene (Verhaltensantizipation)
2. Einstellen des als Nächstes einzuplanenden bzw. auszuführenden Teilplanes $\pi^*(s)$ der geregelten Ebene (Verhaltenssteuerung)
3. Einplanen bzw. Ausführen des eingestellten Teilplanes $\pi^*(s)$ auf der geregelten Ebene (Verhaltensplanung/Ausführung)
4. Erfassen und Klassifizieren des ausgeführten Teilplanes inkl. Vor- und Nachbedingung der geregelten Ebene auf der regelnden Ebene (Verhaltensrückführung)

Die Tabelle 5.2 fasst die Konzeption zur Integration von Verhaltensreglung in die hierarchische Verhaltensplanung zusammen (vgl. hierzu auch Abs. 2.2.1.2 und insb. Tab. 2.1).

Verhaltensregelung	(rollierende) hierarchische Verhaltensplanung
Führungsgröße	Theor. min. Kostenwert des Zielsystems $C'(x)$
Regler	Verhaltensplanung (Top-Ebene, $M^T(\Sigma^{Ext}, RTDP, I_t^T)$)
Stellgröße	(Teil-)Plan (Top- zur Basis-Ebene, π^*)
Regelstrecke	Verhaltensplanung (Basis-Ebene, $M^B(\pi^*, I_{t'}^B)$)
Störgröße	Externe Ziele/Umgebungseinflüsse
Regelgröße	Tats. Kostenwert $C(x)$ des Zielsystems

Tabelle 5.2: Konzeption zur Integration von Verhaltensreglung in hierarchische Verhaltensplanung

Abschließend bleibt noch zu erwähnen, dass die Verhaltensplanung einer überlagerten Basis-Ebene stets revidieren muss, ob der ihr vorgegebene (Teil-)Plan bei aktueller Situation und Zielvorgabe auch umsetzbar bzw. mit einplanbar ist. Sollte dieses nicht der Fall sein, dann wird eigenständig eine Neuplanung auf der Basis-Ebene durchgeführt. Genau an dieser Stelle findet die eigentliche Verhaltensanpassung des kognitiven mechatronischen Systems an sich verändernde Umgebungseinflüsse statt. (Teil-)Pläne, die in der Vergangenheit noch geringe Kosten aufwiesen, können aktuell schon nicht mehr anwendbar sein, um ein Ziel zu erreichen. Die neu

generierten Pläne werden nach ihrer Ausführung erfasst und klassifiziert und reichern dadurch den (erweiterten) adaptiven MDP mit aktuellen Informationen an. Die richtige Wahl des Zeitfensters (vgl. Abs. 5.1) bzw. die Größe des (erweiterten) adaptiven MDPs kann hierbei entscheidend für die Reagibilität der Verhaltensanpassung des kognitiven mechatronischen Systems sein.

6 Validierung

Das Ziel dieses Kapitels besteht darin, die in Kapitel 5 konzipierten Methoden zur Umsetzung der Verhaltensantizipation und -regelung kognitiver mechatronischer Systeme bei langfristiger Planung und Ausführung mittels Simulation unter Verwendung eines Anwendungsmodells experimentell zu validieren. Zu diesem Zweck wird für ein repräsentatives mechatronisches System auf das im SFB 614[249] entwickelte RailCab-System[250], ein innovatives sowie modulares Bahnsystem, bestehend aus autonomen Schienenfahrzeugen (RailCabs) für den Personen- und Gütertransport (vgl. Abb. 6.1), zurückgegriffen und an dazu bereits existierenden Arbeiten angeknüpft. Als konkretes Beispiel für einen Anwendungsfall, im Hinblick auf die langfristige Planung und Ausführung eines Funktionsmoduls, dient das von Schlautmann[251] eingeführte aktive Feder-Neigesystem eines einzelnen RailCabs.[252]

Abbildung 6.1: Versuchsanlage Neue Bahntechnik Paderborn / RailCab im Maßstab 1:2,5 an der Universität Paderborn sowie Versuchsfahrzeuge (RailCab) Gesamtlänge 530m (Quelle: [ADG+09])

[249] Sonderforschungsbereich 614 - Selbstoptimierende Systeme des Maschinenbaus, vgl. [SFB04] und `http://www.sfb614.de/`.

[250] Vgl. Neue Bahntechnik Paderborn (NBP): `http://www.railcab.de/`; zur detaillierten Beschreibung der Funktionsweise vgl. auch [ADG+09].

[251] Vgl. [Sch06].

[252] Vgl. auch hierzu [KAA12] und [AEH+11].

Weil für die anwendungsnahe Simulation des langfristigen Verhaltens ein möglichst präzises Modell des Problembereichs zu verwenden ist, wird zunächst das Anwendungsmodell für das RailCab am Beispiel des aktiven Feder-Neigesystems aufgestellt. In diesem Zusammenhang sind die Beschreibung sowie Implementierung zum Modell der Systemumgebung und zum Modell des aktiven Feder-Neigesystems Gegenstand näherer Betrachtung (vgl. Abs. 6.1). Im Anschluss folgt die Durchführung von Experimenten zur Validierung der Konzeption. Dabei wird insb. das langfristige Verhalten des Systems, sowohl mit als auch ohne Verhaltensantizipation und -regelung, fokussiert und gegenübergestellt. Die Ergebnisse der Experimente werden abschließend dargestellt und evaluiert (vgl. Abs. 6.2).

6.1 Anwendungsfall RailCab – Beispiel: Aktives Feder-Neigesystem

Die Aufgabe des aktiven Feder-Neigesystems besteht im Erhöhen des Komforts für Passagiere während der Fahrt mit einem RailCab. Dazu werden die Aufbaubewegungen durch Kompensation von Streckenanregungen reduziert. Dabei verzichtet das Feder-Neigesystem auf passive Dämpferelemente, zumal diese die vom Streckennetz verursachten Störbewegungen auf den Aufbau übertragen. Der Aufbau ist daher lediglich über Federn mit dem Fahrgestell verbunden. Die zur Kompensation erforderlichen Kräfte werden gezielt durch Hydraulikzylinder erzeugt, die an den Befestigungselementen der Federn verbaut sind und diese in Abhängigkeit der gegenwärtigen Aufbaubewegung beeinflussen.[253]

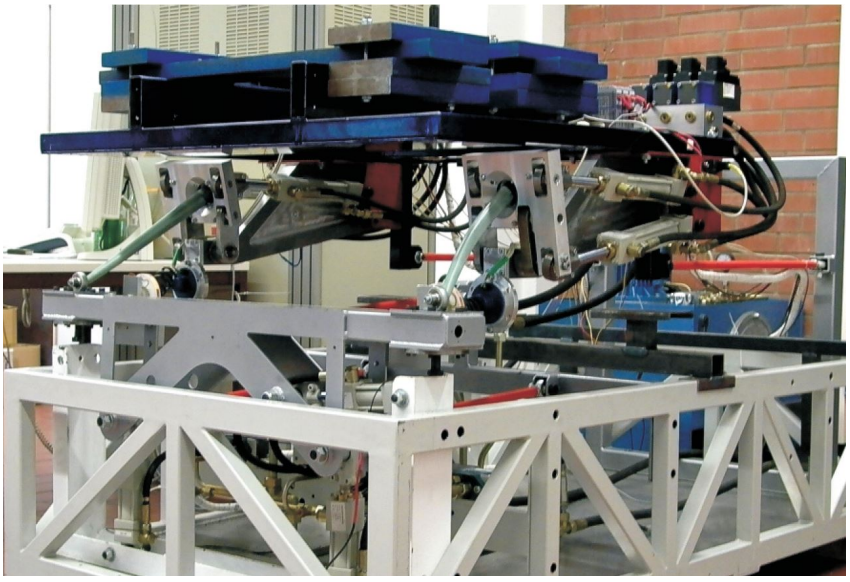


Abbildung 6.2: Prüfstand zur Schienen-Unterflur-Federung (SUrF) (Quelle: [Sch06])

[253] Vgl. [Sch06] und [ADG+09].

Zur Durchführung von Funktionstests wurde im SFB 614 ein Halbfahrzeugprüfstand des aktiven Feder-Neigesystems entwickelt und aufgebaut (vgl. Abb. 6.2). Dieser soll es vorab ermöglichen, das Verhalten des aktiven Feder-Neigesystems bei der Anwendung von verschiedenen Optimierungs- sowie Planungsverfahren zu erproben.[254] Das System unterscheidet dabei zwischen verschiedenen Betriebsarten.[255] Das Einstellen der Betriebsart bzw. der Ausprägung davon (Operationsmodus, vgl. Abs. 2.1.1) entscheidet zum einen über den Kompensationsgrad der Aufbaubewegung bzw. den Grad des Fahrtkomforts und zum anderen über den dafür erforderlichen Energieverbrauch des Funktionsmoduls. Die Kompensation der Aufbaubewegungen geht mit einer Erhöhung des Energieverbrauchs des aktiven Feder-Neigesystems für den verstärkten Betrieb der Hydraulikzylinder einher. Für einen optimalen Gesamtbetrieb des Funktionsmoduls ist daher ein Mehrzieloptimierungsproblem mit den beiden gegensätzlichen Zielen „minimiere Aufbaubewegungen“ und „minimiere Energieverbrauch“ zu lösen. Weil Mehrzieloptimierung aufwendig ist, werden für charakteristische Streckenabschnittstypen die Lösungen in Form von Pareto-Mengen im Vorfeld bestimmt und für die Betriebszeit zur Auswahl bereitgestellt.[256] Genau diese Lösungen stellen die wählbaren Operationsmodi bzw. Aktionen für die Planung dar. Hier besteht die Aufgabe der Planung darin, für einen festgelegten Streckenverlauf proaktiv eine Folge von Operationsmodi für die spätere Ausführung einzuplanen, die einen bestmöglichen Betrieb des aktiven Feder-Neigesystems gewährleistet. Dadurch kann z. B. auf ohnehin „glatten“ Streckenabschnitten Energie eingespart und für spätere „rauere“ Streckenabschnitte eingesetzt werden. Da nicht nur die Streckenbeschaffenheit, sondern auch die Umgebungseinflüsse, wie beispielsweise Gegen- oder Seitenwinde, den Aufbau des RailCabs anregen, handelt es sich um ein probabilistisches Planungsproblem (vgl. Abs. 2.1.3.2), bei dem diese Faktoren mitzuberücksichtigen sind.[257]

In den nächsten beiden Abschnitten werden zunächst das Modell der Systemumgebung (vgl. Abs. 6.1.1) und das Modell des RailCabs mit Fokus auf dem aktiven Feder-Neigesystem (vgl. Abs. 6.1.2) vorgestellt.

6.1.1 Modell der Systemumgebung

In diesem Abschnitt findet die Modellierung der Systemumgebung statt. Diese zielt insb. auf die Simulation von langfristiger Planung und Ausführung mehrerer Aufträge unter Berücksichtigung von Nichtdeterminismus und (quasi-)kontinuierlichen

[254] Vgl. [ADG+09], S. 66 ff.

[255] 1) Passiv: Das Federungssystem ist deaktiviert, die einzige Dämpfung besteht in der Strukturdämpfung des Fahrzeugs; 2) Aktiv: Die aktive Federung induziert Zusatzkräfte zur Dämpfung der Bewegung zwischen Aufbau und Fahrwerk bzw. zur Dämpfung der Absolutbewegung des Aufbaus; 3) Aktiv mit Störgrößenkompensation: Zusätzlich zur aktiven Bedämpfung der Aufbaubewegung werden bekannte – von der streckenseitigen Informationsverarbeitung abgerufene – Störungen kompensiert, sodass sie nicht auf den Aufbau wirken können. (Quelle: [ADG+09])

[256] Vgl. [VT08].

[257] Vgl. [KAA12], [KSWR12] und [AEH+11].

Prozessen ab. Zudem wird das konkrete Planungsszenario aus Sicht der Top-Ebene (vgl. Abs. 2.2.1.1) betrachtet.

Die Abbildung 6.3 skizziert beispielhaft das Modell der Systemumgebung in Form eines Streckennetzes, bestehend aus miteinander verbundenen Haupt- sowie Zwischenstationen. Die Verbindungen der (Haupt-)Stationen sind durch Streckenabschnitte (engl. *track sections*, kurz TSC) modelliert, die vom RailCab befahren werden können. Streckenabschnitte zwischen zwei Hauptstationen bilden zusammen eine einzelne Strecke (engl. *track*). Damit die Streckenabschnitte hinsichtlich der im Vorfeld zur Verfügung zu stellenden Pareto-Lösungen differenzierbar sind, repräsentieren diese jeweils unterschiedliche Streckenabschnittstypen (z. B. verschiedene „glatte“ oder „raue“ Streckenabschnittstypen). In dieser Arbeit wird insgesamt zwischen genau 10 Streckenabschnittstypen (I-X) unterschieden, zu denen im späteren Verlauf noch die entsprechenden Pareto-Lösungen bekannt gegeben werden.

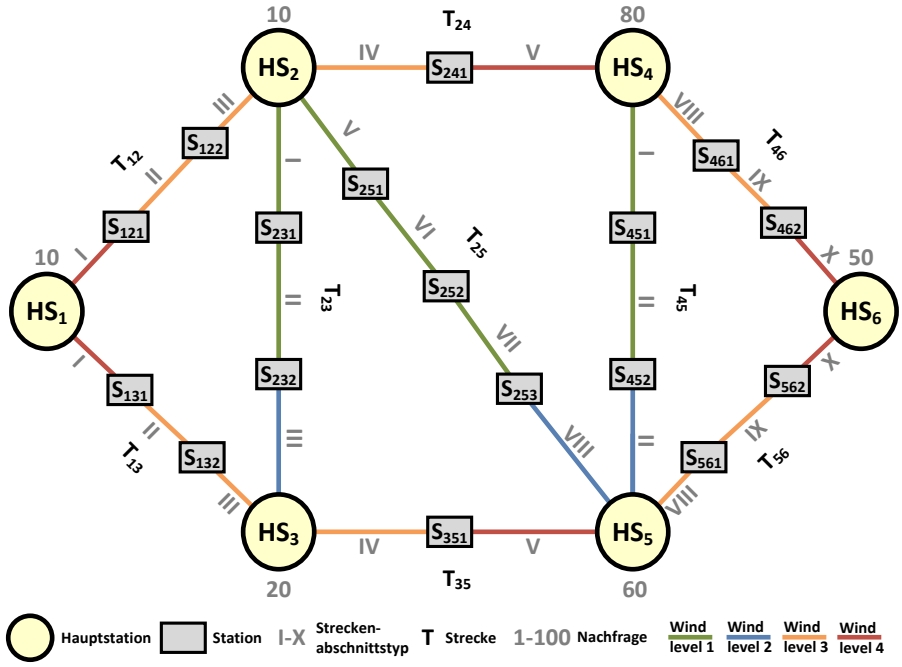


Abbildung 6.3: Skizze vom Modell der Systemumgebung: Streckennetz mit (Haupt-) Stationen, Streckentypen, Windstärken und Nachfragegewichtung

Damit zur Validierung des Konzeptes die zwei wesentlichen Umgebungseigenschaften, der Nichtdeterminismus und die kontinuierlichen Prozesse, simuliert werden können, ist jeder Streckenabschnitt zusätzlich einem von vier möglichen Windverhältnissen zugeordnet (vgl. Abb. 6.3). Mit dem Verwenden einer stetigen Wahrscheinlichkeitsverteilung für die Windverhältnisse und einem entsprechenden Zufallsgenerator ist so zum einen der Wert einer nichtdeterministischen Größe in Form von Gegen- oder Seitenwind generierbar und zum anderen eine von dieser indirekt

beeinflusste (quasi-)kontinuierliche Zustandsgröße des RailCabs umsetzbar.

Als stetige Wahrscheinlichkeitsverteilung wurde auf die Weibull-Verteilung mit bestimmten Skalierungsfaktoren und Formparametern als Basis zur Generierung der vier unterschiedlichen Windverhältnisse in Metern pro Sekunde (m/s) zurückgegriffen (vgl. Abb. 6.4). In dem hier beschriebenen Modell wird in vereinfachter Weise angenommen, dass die generierte Windstärke, umgerechnet in Kilometer pro Stunde (km/h), proportional die Aufbaubewegung verstärkt sowie den Energieverbrauch des aktiven Feder-Neigesystems erhöht. Dieses geschieht hier jeweils um den Faktor $wind\frac{km/h}{100}$.

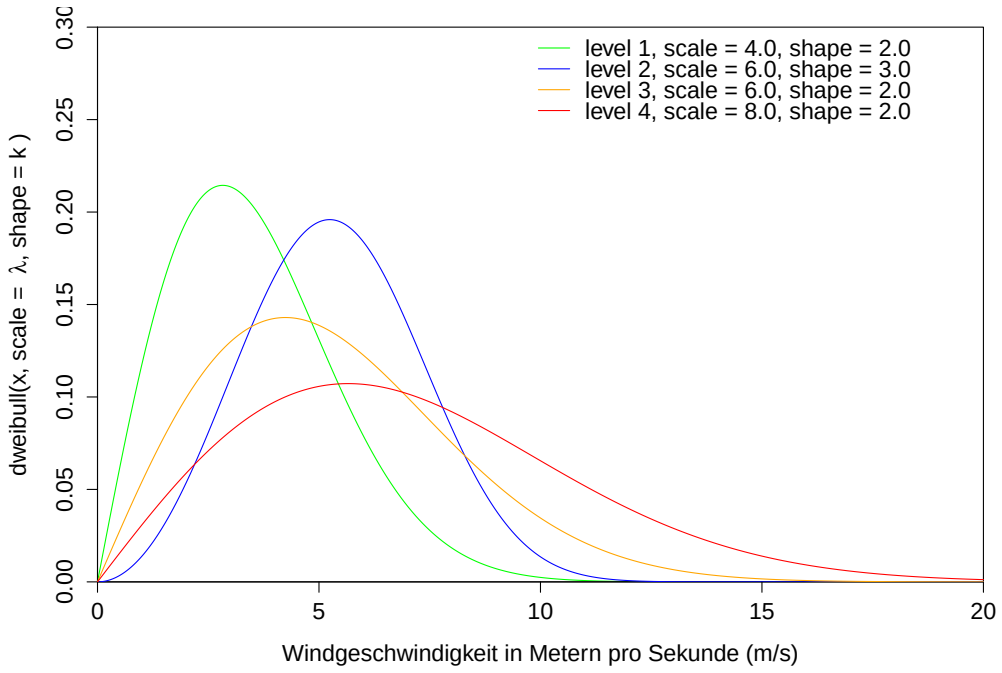


Abbildung 6.4: Dichtefunktionen der Weibullverteilung mit unterschiedlichen Skalierungsfaktoren λ und Formparametern k zur Simulation von Windgeschwindigkeiten in Metern pro Sekunde

Der Einzelauftrag eines RailCabs besteht in diesem Modell aus der Fahrt auf einer Strecke zwischen zwei Hauptstationen, die wie zuvor beschrieben, das Berücksichtigen mehrerer Streckenabschnitte mit unterschiedlichen Streckenabschnittstypen und Windverhältnissen umfasst.

Zum Erzeugen von Aufträgen wird ein Kunde simuliert, der ausgehend von der aktuellen Hauptstation des RailCabs eine Zielstation zufällig in Abhängigkeit von den Nachfragegewichtungen (vgl. Abb. 6.3) generiert. Zwischen Startstation und generierter Zielstation wird dann mittels des Dijkstra-Algorithmus der kürzeste Weg bestimmt (eine Differenzierung der Streckenlänge, wie beispielsweise in Kilometer, findet in diesem vereinfachten Modell nicht statt). Die ermittelte Route von zu befahrenden Strecken bzw. abzufahrenden Hauptstationen wird im Anschluss als

Sequenz von Einzelaufträgen an das RailCab zur Planung und Ausführung übergehen.

Aus Sicht der überlagerten Top-Ebene besteht hier das längerfristige Ziel der Planung darin, die Einzelaufträge in Summe mit minimalem Energieverbrauch zu absolvieren. Dabei wird stets die Erfüllbarkeit des aktuellen Einzelauftrags auf der überlagerten Basis-Ebene, die der Planung des aktiven Feder-Neigesystems entspricht, berücksichtigt (vgl. Abs. 5.3).

Der Abbildung 6.5 ist das entsprechende Klassendiagramm zum Modell der Systemumgebung zu entnehmen.

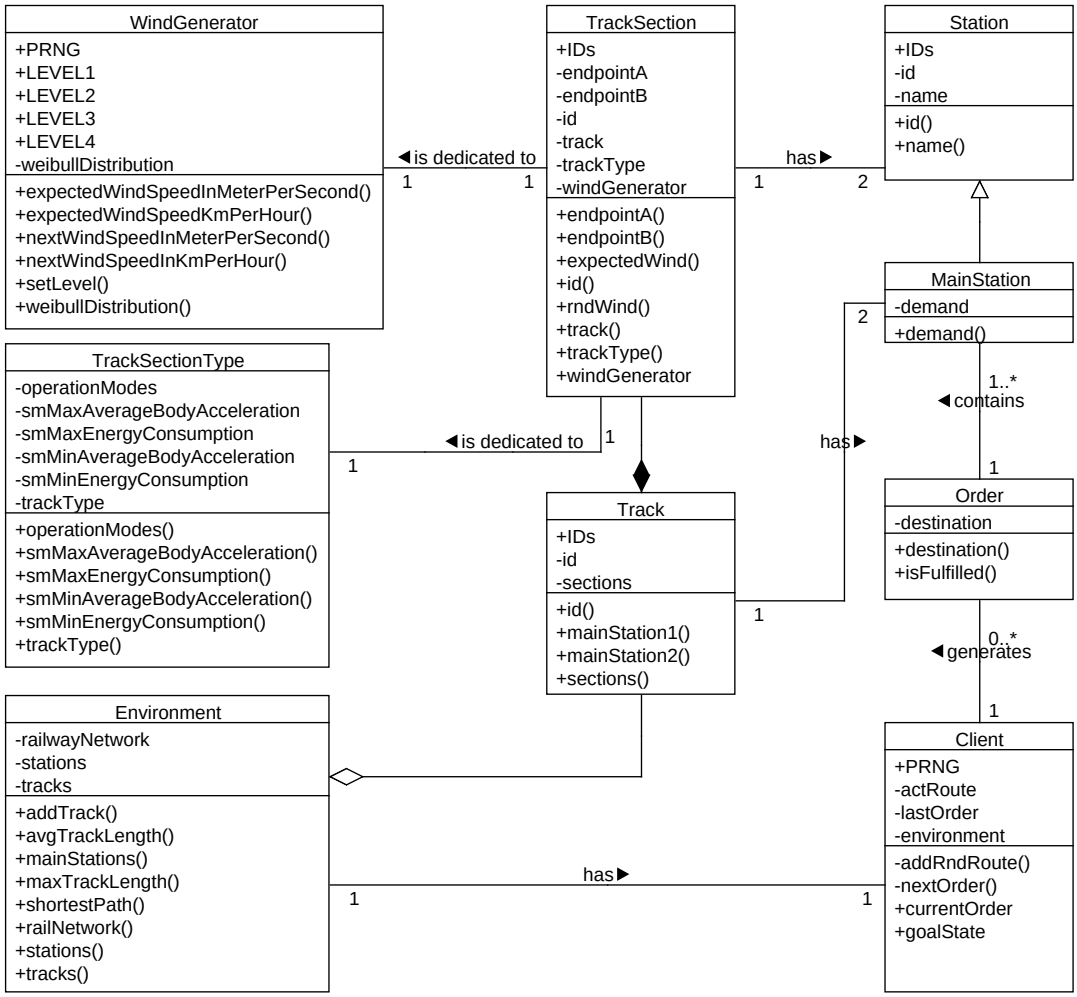


Abbildung 6.5: Klassendiagramm zum Modell der Systemumgebung

6.1.2 Modell des RailCabs – Beispiel: Aktives Feder-Neigesystem

In diesem Abschnitt wird das Modell des RailCabs am Beispiel des aktiven Feder-Neigesystems aufgestellt und in das Modell der Systemumgebung (vgl. Abs. 6.1.1) integriert. Das aktive Feder-Neigesystem verfolgt als eigenständiges Funktionsmodul des RailCabs zwei gegensätzliche Ziele: Die Minimierung der Aufbaubewegung (gleichzusetzen mit der Maximierung des Fahrkomforts) und die Minimierung des Energieverbrauchs. Das aufgrund der Komplexität schon während der Entwurfszeit des kognitiven mechatronischen Systems zu lösende multikriterielle Optimierungsproblem zur Berechnung der optimalen Operationsmodi für die Streckenabschnitte der Systemumgebung (vgl. Abs. 6.1.1) beinhaltet daher die Verknüpfung zweier Zielfunktionen $f_1(t)$ und $f_2(t)$ mit folgenden Ausdrücken:[258]

$$f_1(t) = \frac{1}{t - t_0} \cdot \int_{\tau=t_0}^{t} \sum_i (W_i(a_i(\tau)))d\tau \tag{6.1}$$

$$f_2(t) = \frac{1}{t - t_0} \cdot \int_{\tau=t_0}^{t} P_{hydr}(\tau)d\tau \tag{6.2}$$

Während der Ausdruck 6.1 die gewichtete mittlere Aufbaubeschleunigung für die Vertikale (i=1), Horizontale (i=2) und Fahrtrichtung (i=3) beschreibt, so wird mit dem Ausdruck 6.2 die mittlere hydraulische Leistung berechnet. Die Abbildung 6.6 zeigt beispielhaft die Pareto-Lösungen der multikriteriellen Optimierung für verschiedene Streckenrauigkeiten.

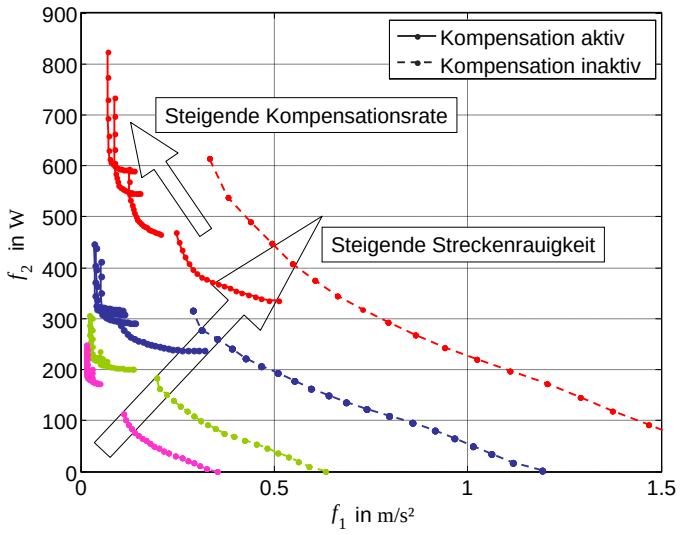


Abbildung 6.6: Paretomengen des aktiven Feder-Neigesystems für unterschiedliche Streckenrauigkeiten (Quelle: [ADG+09])

[258] Zur detaillierten Berechnung vgl. [VT08].

Die berechneten Zielfunktionswerte zu den in dieser Arbeit zur Einplanung auf der Basis-Ebene (vgl. Abs. 2.2.1.1) für die 10 unterschiedlichen Streckenabschnittstypen (I-X, vgl. Abs. 6.1.1) zum optimalen Betrieb des aktiven Feder-Neigesystems verwendeten Operationsmodi sind der Tabelle 6.1 zu entnehmen. Genau diese Werte bilden die Grundlage für die später in dieser Arbeit durchgeführten Experimente zur Validierung der Konzeption. An dieser Stelle sei darauf hingewiesen, dass die Länge eines Streckenabschnitts hier normiert betrachtet wird und sich an der maximalen Geschwindigkeit des Versuchsfahrzeugs (vgl. Abb. 6.2) von $10m/s$ ($= 36km/h$) orientiert. Der Wert von f_2 in Wattsekunden (ws) beispielsweise, entspricht daher in Wirklichkeit der Energieleistung in Watt für eine Fahrtdauer von 1 Sekunde über einen Streckenabschnitt von $10m$.[259]

Operations-modus	**Ziel-funktion**	**Streckenabschnittstyp**									
		I	**II**	**III**	**IV**	**V**	**VI**	**VII**	**VIII**	**IX**	**X**
a	f_1	0,117	0,233	0,350	0,466	0,583	0,699	0,816	0,932	1,049	1,166
	f_2	196	393	589	786	982	1179	1375	1572	1768	1965
b	f_1	0,152	0,304	0,457	0,609	0,761	0,913	1,066	1,218	1,370	1,522
	f_2	165	329	494	659	823	988	1153	1317	1482	1647
c	f_1	0,192	0,385	0,577	0,770	0,962	1,155	1,347	1,540	1,732	1,925
	f_2	142	283	425	567	709	850	992	1134	1275	1417
d	f_1	0,224	0,449	0,673	0,897	1,122	1,346	1,570	1,794	2,019	2,243
	f_2	122	245	367	489	612	734	856	979	1101	1224
e	f_1	0,262	0,523	0,785	1,047	1,308	1,570	1,832	2,093	2,355	2,617
	f_2	104	208	313	417	521	625	730	834	938	1042
f	f_1	0,298	0,595	0,893	1,191	1,488	1,786	2,084	2,381	2,679	2,977
	f_2	87	173	260	346	433	520	606	693	779	866
g	f_1	0,331	0,662	0,994	1.325	1,656	1,987	2,318	2,649	2,981	3,312
	f_2	69	138	206	275	344	413	482	550	619	688
h	f_1	0,375	0,749	1,124	1,499	1,873	2,248	2,623	2,997	3,372	3,747
	f_2	50	99	149	199	248	298	348	398	447	497
i	f_1	0,435	0,870	1,305	1,739	2,174	2,609	3,044	3,479	3,914	4,349
	f_2	27	55	82	110	137	164	192	219	247	274

Tabelle 6.1: Werte für f_1 (gewichtete mittlere Aufbaubeschleunigung in m/s^2) und f_2 (Energieverbrauch in ws) der Operationsmodi aus der multikriteriellen Optimierung des aktiven Feder-Neigesystems (Quelle: [KAA12])

Im Folgenden seien TSC die Menge aller Streckenabschnitte des Typs I-X (vgl. Tab. 6.1) mit den Windverhältnissen 1-4 und ST die Menge aller Stationen aus dem Modell der Systemumgebung (vgl. Abs. 6.1.1). Des Weiteren seien $f_1(om)$ der Zielfunktionswert von f_1 und $f_2(om)$ der Zielfunktionswert von f_2 bei Anwendung des Operationsmodus $om \in OM = \{a, ..., i\}$ (vgl. Tab. 6.1).

[259] Vgl. [AEH+11] und `http://www.railcab.de/`.

Zur Formulierung des Planungsproblems für die Basis-Ebene wird zuallererst die Zustandsrepräsentation des RailCabs im Planungsmodell definiert. Diese ist hier durch den Zustandsvektor $\vec{s} = (st, soc, ltsc)$ mit den Zustandsvariablen $st \in ST$ für den aktuellen Standort, *soc* für die verfügbare Energie (engl. *State of Charge*) in Wattsekunden und $ltsc \in TSC$ für den zuvor befahrenen Streckenabschnitt des RailCabs gegeben. Letzteres soll insb. ermöglichen, feststellen zu können, aus welcher Richtung sich das RailCab einem Standort angenähert hat.

Unter Berücksichtigung dieser Zustandsrepräsentation werden die zwei vom RailCab ausführbaren Aktionen DRIVE und LOAD zum Modell hinzugefügt. Die Aktion DRIVE beschreibt die Fahrt des RailCabs über einen Streckenabschnitt und das gleichzeitige Anwenden eines für das aktive Feder-Neigesystem gewählten Operationsmodus (z. B. für einen Streckenabschnitt des Typs V, der Operationsmodus d mit $f_1(d) = 1,122\ m/s^2$ Aufbaubewegungen und $f_2(d) = 612\ ws$ Energieverbrauch, vgl. Tab. 6.1). Als Parameter für den zu befahrenden Streckenabschnitt *tsc* werden der Aktion DRIVE die Startstation *start*, die Zielstation *destination* sowie der festgelegte Operationsmodus *om* übergeben. Die Übergabe der Start- und Zielstation kann dabei, je nach Betrachtung des Streckenabschnitts, auf zwei Arten erfolgen (Vertauschung der Reihenfolge) und definiert somit die Fahrtrichtung des RailCabs. Der Effekt der Aktion DRIVE auf den Zustand des RailCabs besteht hier in der Aktualisierung der Zustandsvariablen mit $st = destination$, $soc = soc - f_2(om)$ und $ltsc = tsc$. Dieser Zustandsübergang beschreibt allerdings nur den rein deterministischen Effekt der Aktion DRIVE. Weil der nichtdeterministische Einfluss der Systemumgebung, gegeben durch die Windverhältnisse der Streckenabschnitte (vgl. Abs. 6.1.1), mit in die Planung einzubeziehen ist, wird daher die verfügbare Energie in der probabilistischen Erweiterung des Modells durch $soc = soc - f_2^{exp}(om)$ aktualisiert mit

$$f_2^{exp}(om) = f_2(om) + f_2(om) * \frac{E(wind_{tsr}\frac{km}{h})}{100} \tag{6.3}$$

für den zu erwartenden Energieverbrauch von Operationsmodus *om* in Abhängigkeit des Erwartungswertes für die Windgeschwindigkeit in km/h auf dem zu befahrenden Streckenabschnitt *tsr* (vgl. Abs. 6.1.1). Eine gültige Vorbedingung der probabilistischen Aktion DRIVE ist dabei durch $st = start$ und $soc \geq f_2^{exp}(om)$ gegeben. Zu jeder Kombination aus Streckenabschnitt, Operationsmodus und Fahrtrichtung existiert eine dieser Aktionen. Die Planungsinstanz zum Beispielmodell aus Abb. 6.3 würde insgesamt $26 * 10 * 2 = 520$ mögliche DRIVE-Aktionen beinhalten.

Die Aktion LOAD hingegen beschreibt den Vorgang zum Aufladen der Energiereserve des RailCabs, zumal die Aktion DRIVE im gegebenen Modell fortwährend Energie verbraucht und aufgrund ihrer Vorbedingung nach einiger Zeit zum Liegenbleiben des RailCabs führt. Der Aktion LOAD wird als Parameter die aktuelle Station übergeben. Der Effekt auf den Zustand des RailCabs besteht hier lediglich in der Aktualisierung der Zustandsvariablen zur Energieverfügbarkeit mit $soc = soc_{max}$, wobei soc_{max} eine im Vorfeld festgelegte Modellkonstante für die maximale Energiekapazität des RailCabs in Wattsekunden darstellt. Zu jeder Sta-

tion existiert eine LOAD-Aktion. Das RailCab kann daher in jeder Station einen Ladevorgang mit einplanen. Die Planungsinstanz zum Beispielmodell aus Abb. 6.3 würde 23 mögliche LOAD-Aktionen beinhalten.

Als Nächstes wird das Ziel der Planung auf der Basis-Ebene für Einzelaufträge formuliert. Im vorliegenden Modell besteht dieses aus der Minimierung des Gesamtenergieverbrauchs bei einem bestimmten durchschnittlichen Mindestkomfort für den eingeplanten Einzelauftrag (vgl. Abs. 6.1.1). Sei OM_{TSC} die Folge von gewählten Operationsmodi zur Erfüllung des aktuellen Einzelauftrags, dann ist die Zielfunktion des Planungsproblems gegeben durch

$$F(OM_{TSC}) = \sum_{om_{tsc} \in OM_{TSC}} f_2^{exp}(om_{tsc}) \longrightarrow min \tag{6.4}$$

mit den Nebenbedingungen

$$F(OM_{TSC}) = \frac{1}{|OM_{TSC}|} \sum_{om_{tsc} \in OM_{TSC}} f_1^{exp}(om_{tsc}) \leq comfort_{min} \tag{6.5}$$

, $soc \geq 0$ und $st = finaldestination$.

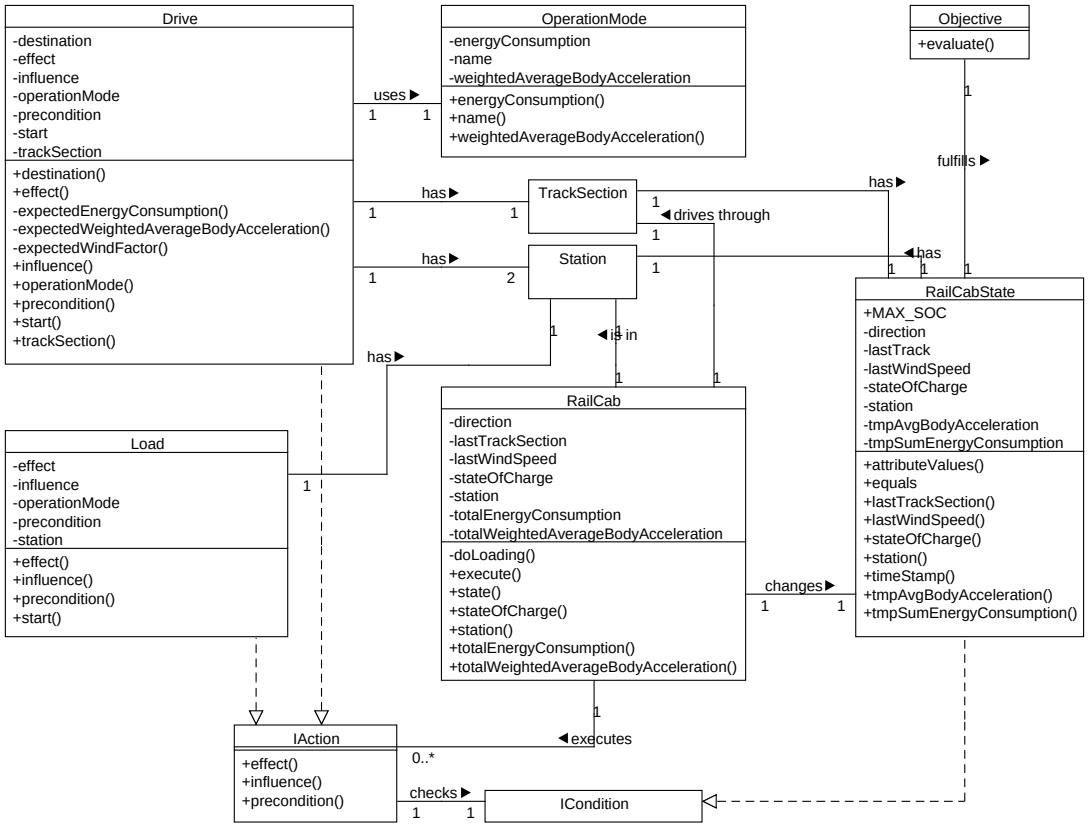


Abbildung 6.7: Klassendiagramm RailCab – Beispiel: Aktives Feder-Neigesystem

Das Klassendiagramm in Abbildung 6.7 zeigt das vollständige Modell des RailCabs am Beispiel des aktiven Feder-Neigesystems. Auf Grundlage der definierten Zustandsrepräsentation, der Aktionen DRIVE und LOAD sowie der formulierten Zielfunktion kann auf der Basis-Ebene für den aktuellen Einzelauftrag eine zustandsbasierte Suche zum Finden der optimalen Folge von Operationsmodi durchgeführt und zur Ausführung an das kognitive mechatronische System übergeben werden.

6.2 Experimente

Ziel dieses Abschnitts ist die Beschreibung und Auswertung der durchgeführten Experimente zur Validierung des in Kapitel 5 entwickelten Konzeptes zur Verhaltensantizipation und -regelung von kognitiven mechatronischen Systemen bei langfristiger Planung und Ausführung.

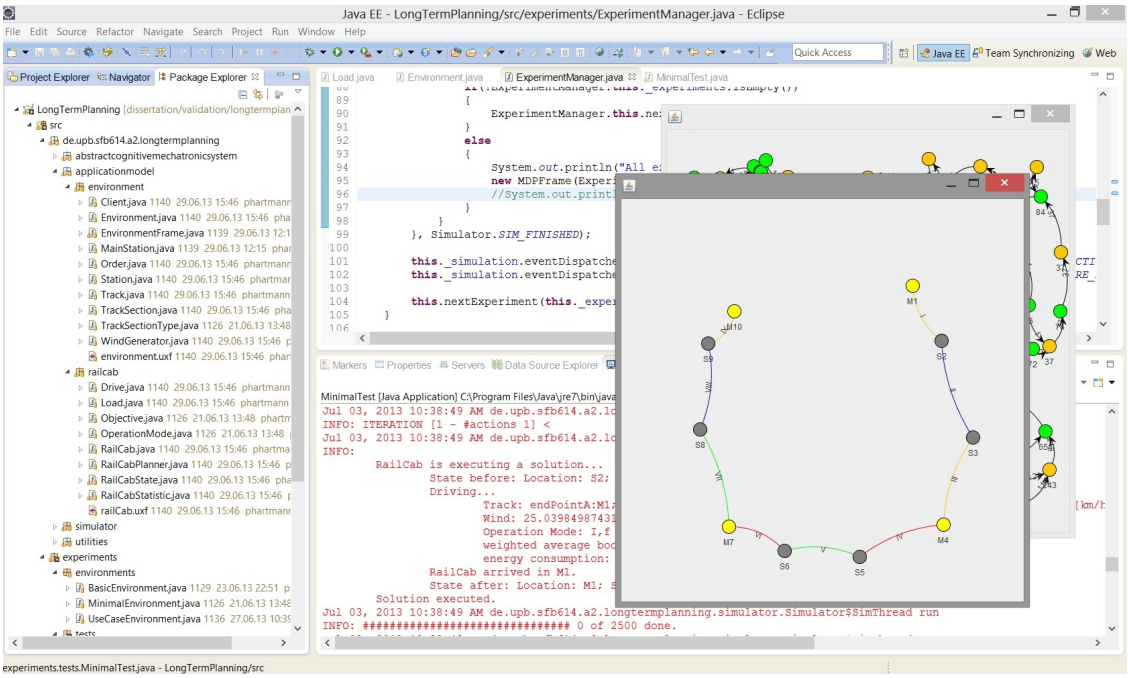


Abbildung 6.8: Simulationsplattform in Eclipse zur Validierung der Verhaltensantizipation und -regelung

Als Grundlage der Experimente diente der in Abschnitt 6.1 beschriebene Anwendungsfall RailCab am Beispiel des aktiven Feder-Neigesystems. Dabei wurde insb. auf das Modell der Systemumgebung (vgl. Abs. 6.1.1) und das Modell des RailCabs (vgl. Abs. 6.1.2) zurückgegriffen und eine entsprechende Simulation umgesetzt. Zu diesem Zweck fand die Implementierung einer Simulationsplattform sowie

der beschriebenen Modelle in der Programmiersprache Java[260] unter Verwendung der Entwicklungsumgebung Eclipse[261] statt (vgl. Abb. 6.8).

Während der Durchführung der Experimente erfolgte innerhalb der hierarchischen Verhaltensplanung und -regelung folgende Zuteilung von Planungsebenen (vgl. hierzu auch Abs. 5.3):

- Basis-Ebene: Planung von Einzelaufträgen mit dem (kurzfristigen) Ziel den Energieverbrauch zu minimieren und dabei einen durchschnittlichen Mindestkomfort sicherzustellen
- Top-Ebene: Auftragsübergreifende Planung mit dem (langfristigen) Ziel den Energieverbrauch über alle Einzelaufträge insgesamt zu minimieren

Abgesehen von der reinen Validierung der Funktionalität und Wechselwirkungen der Planungsebenen sollte zudem folgende Annahme überprüft werden: Die aufgrund des begrenzten Planungshorizontes der Basis-Ebene und der damit einhergehenden Einplanung von aus langfristiger Sicht ggf. suboptimalen Folgen an Operationsmodi zu viel verursachter Energieverbrauch kann durch die Instruktionen der übergreifenden Planung der Top-Ebene reduziert bzw. kompensiert werden.

Die nächsten beiden Abschnitte dokumentieren die zwei durchgeführten Experimente: Zum einen zum Grundverhalten in einer Beispielumgebung (Experiment I, vgl. Abs. 6.2.1) und zum anderen zum Verhalten in unterschiedlichen Umgebungen (Experiment II, vgl. Abs. 6.2.2). Beide Experimente umfassen eine detaillierte Beschreibung sowie Darstellung und Evaluation der Ergebnisse.

6.2.1 Experiment I: Grundverhalten in einer Beispielumgebung

6.2.1.1 Ziel und Beschreibung des Experiments

Ziel von Experiment I war es, das Grundverhalten des RailCabs bei langfristiger Planung und Ausführung anhand des Modells zum aktiven Feder-Neigesystem (vgl. Abs. 6.1.2) im Zusammenschluss mit dem Modell einer Beispielumgebung (vgl. Abs. 6.1.1) zu untersuchen. Dabei sollte insb. auch die Funktionalität der im Konzept entwickelten Methoden zur Erfassung und Klassifikation von Plänen (vgl. Abs. 5.1), Verhaltensantizipation (vgl. Abs. 5.2) sowie Integration von Verhaltensregelung in die hierarchische Verhaltensplanung (vgl. Abs. 5.3) getestet werden.

Dazu wurde das in Abschnitt 6.1.1 beschriebene Beispielmodell der Systemumgebung mithilfe der fertiggestellten Simulationsplattform implementiert[262] (vgl. Abb. 6.9). Im Anschluss folgte die Simulation zur Planung und Ausführung von 10.000 Aufträgen sowie die Auswertung des Verhaltens vom RailCab bzgl. des Energieverbrauchs, der Aufbaubewegungen, der Anzahl von Ausfällen und der Anzahl von

[260] Vgl. `http://www.java.com/de/`.

[261] Vgl. `http://www.eclipse.org/`.

[262] Die Graphische Visualisierung des Streckennetzes aus Abbildung 6.9 basiert auf der Java-Bibliothek JUNG (Java Universal Network/Graph Framework, vgl. `http://jung.sourceforge.net`).

Ladezyklen.

Zum Vergleich bzw. zur Bewertung fand die Simulation und Gegenüberstellung von drei unterschiedlichen Planungsmethoden statt:

1. Anschlussplanung (AP)
2. Ereignisorientiert angestoßene rollierende Planung (RP)
3. Ereignisorientiert angestoßene rollierende Planung mit Verhaltensantizipation und -regelung (hierarchische Planung, HP/RTDP)

Weil das Verhalten des RailCabs auch für unterschiedliche Ausgangssituationen bzw. Verläufe von Umfeldeinflüssen zu prüfen war, wurde dieser Vorgang in der identischen Beispielumgebung mit 20 verschiedenen Startwerten (engl. *seeds*) für den verwendeten Zufallsgenerator wiederholt und die Ergebnisse dazu im gleichen Maße dokumentiert.

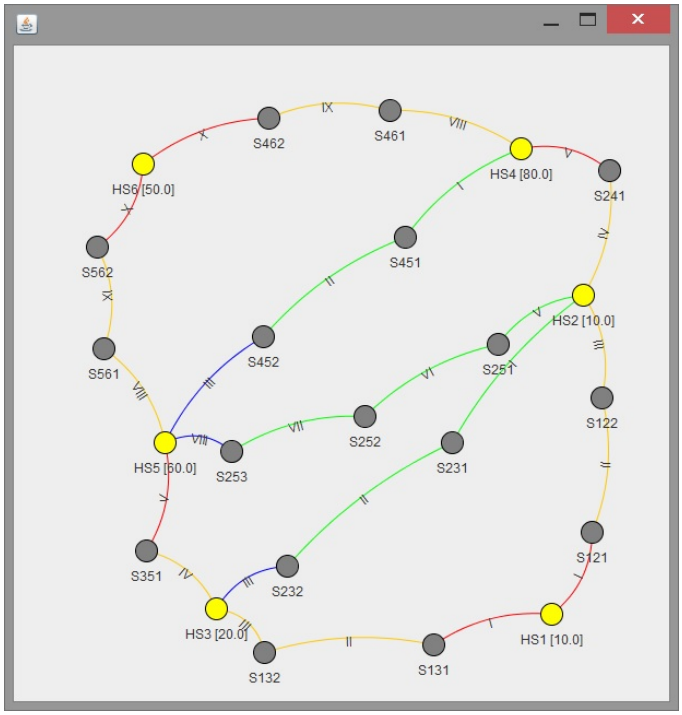


Abbildung 6.9: Graphische Ausgabe des Streckennetzes mit (Haupt-)Stationen, Streckenabschnittstypen und Windverhältnissen

Im weiteren Verlauf sollen konkrete Angaben zur Initialisierung und Parametrierung der verwendeten Methoden aus Kapitel 5 für die durchgeführte Simulation von Verhaltensantizipation und -regelung gemacht werden.

Als initiale Klassenschwerpunkte zur Klassenbildung der Zustände (vgl. Abs. 5.1.2) wurden repräsentative Zustände in Abhängigkeit des vorliegenden Streckennetzes gewählt. Zu jeder Kombination aus (Haupt-)Station, bestimmter Energieverfügbarkeit und Fahrtrichtung erfolgte das Erzeugen der initialen Klassenschwerpunkte

$$\vec{m} = (st = st_i, soc = soc_j * SOC_{max}, ltsc = tsc_{ik}) \tag{6.6}$$

mit $st_i \in ST$, $soc_j \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ für die Energieverfügbarkeit in Prozent und $tsc_{ik} \in TSC_i$, wobei TSC_i die Menge der eingehenden Streckenabschnitte einer Station i darstellt. Eindeutige sowie fortlaufende Nummerierungen (IDs) der Stationen und Streckenabschnitte stellten dabei die Repräsentation im reellen Zahlenraum für die erforderlichen Zustandsvektoren in der hier durchgeführten partitionierenden Clusteranalyse sicher (vgl. Abs. 5.1.2.3). Das Modell aus Abbildung 6.9 verfügte damit initial über 312 Zustandsklassen (vgl. hierzu auch Abs. 6.1.1 und 6.1.2).

Um die drei gegebenen Zustandsdimensionen Station st, Energieverfügbarkeit soc und Streckenabschnitt tsc während der Klassifikation mitzuberücksichtigen, wurden für die Parametrierung des Proximitätsmaßes zu den Funktionen F, Gewichtungswerten W und Normalisierungsfaktoren U die Mengen

$$F = \{abs(x), abs(x), abs(x)\},\ W = \{0.65, 0.10, 0.25\} \text{ und}$$
$$U = \left\{\tfrac{1}{|ST|}, \tfrac{1}{SOC_{max}}, \tfrac{1}{|TSC|}\right\}$$

festgelegt (vgl. Abs. 5.1.2.2). Während die Klassifizierung der Pläne automatisch nach jedem Planungszyklus durchgeführt wurde, erfolgte die Aktualisierung der Klassenbildung mithilfe des partitionierenden Clusterings nur alle 100 Planungszyklen.

Die verhaltens- und ergebnisorientierte Echtzeit-Antizipation (Top-Ebene) zum Ermitteln der kostenminimalen Instruktionen für die kurzfristige Planung des RailCabs (Basis-Ebene) wurde auf dem erweiterten adaptiven MDP (vgl. Abs. 5.1.2.1 und Abb. 6.10) mit 250 Pfaden (*trails*) pro Durchlauf ausgeführt (vgl. Abs. 5.2.2.4). Hierbei wurde die Kostenfunktion zum Bestimmen der erwarteten direkten Kosten in den Zustandsklassen mit

$$F = \{abs(x), abs(x), abs(x)\},\ W = \{0, 1, 0\} \text{ und } U = \{0, 1, 0\}$$

parametriert (vgl. Abs. 5.2.1.1). Weil aufgrund dieser Parametrierung bzw. Gewichtung in die Kostenbewertung der Pläne nur die Dimension der Energieverfügbarkeit soc mit einfloss, war daher das ausschließliche Einbeziehen der direkten Kosten für Energie bei der langfristigen Planung der Top-Ebene möglich. Die maximale Energiekapazität soc_{max} des RailCabs wurde dabei auf den durchschnittlichen maximal zu erwartenden Energieverbrauch pro Streckenabschnitt, multipliziert mit der durchschnittlichen Anzahl von Streckenabschnitten pro Strecke, festgelegt.

Als Planabstand für die beiden ereignisorientiert angestoßenen rollierenden Planungsverfahren (RP und HP/RTDP) wurde ein Streckenabschnitt bzw. Operationsmodus, der genau einer ausgeführten Aktion entspricht, gewählt. Nach jedem befahrenen Streckenabschnitt wurde daher bei diesem Verfahren ein vollständiger Zyklus bestehend aus Erfassung und Klassifikation, Verhaltensantizipation, Verhaltensregelung, Verhaltensplanung und Ausführung (vgl. Abb. 5.1) angestoßen.

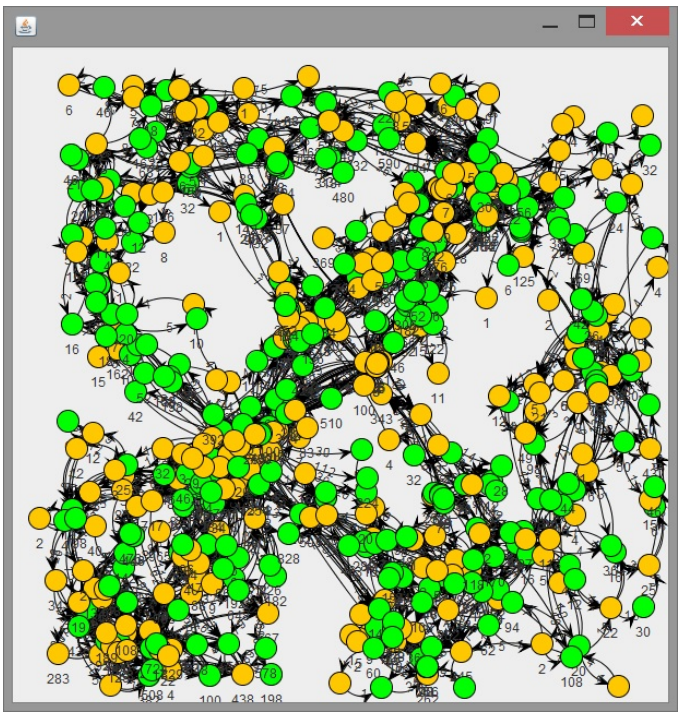


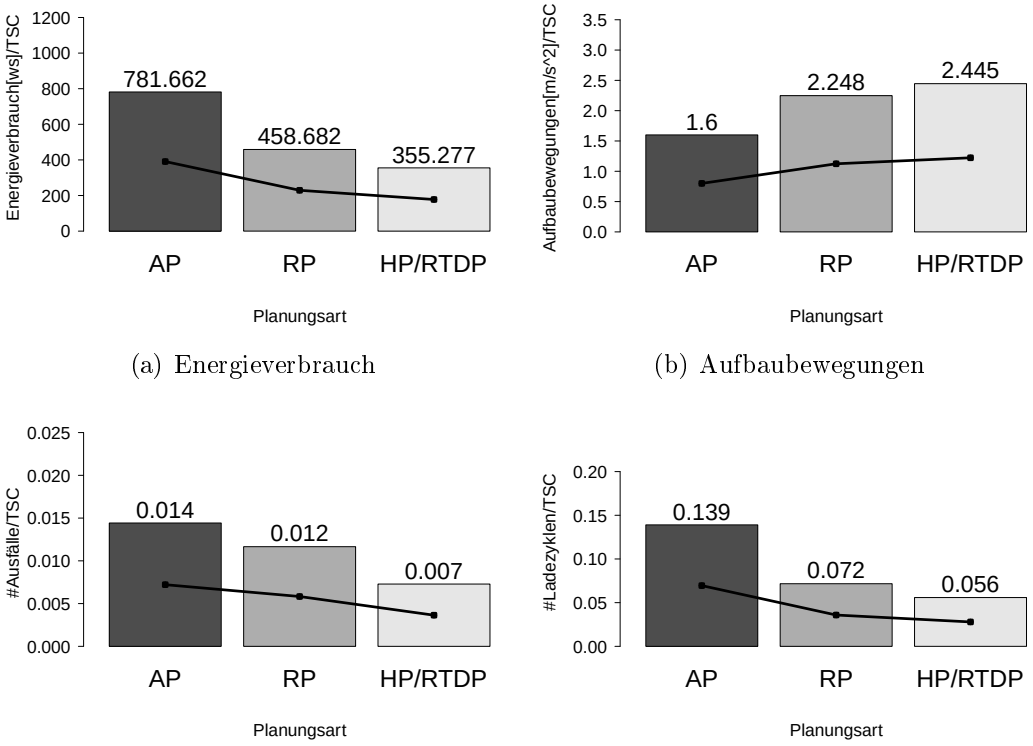
Abbildung 6.10: Graphische Ausgabe des (erweiterten) adaptiven MDPs nach 10.000 simulierten Aufträgen mit Zustandsklassenknoten (grün), Planknoten (orange) und Kanten (schwarz)

6.2.1.2 Ergebnisse und Evaluation

Den Abbildungen 6.12 bis 6.15 sind die Ergebnisse der zum Experiment I durchgeführten Simulation, einem Durchlauf des Beispielmodels mit zunächst nur einem Startwert, in Form von Charts über die Entwicklung des Energieverbrauchs, der Aufbaubewegungen, der Anzahl von Ausfällen und der Anzahl von Ladezyklen im Durchschnitt pro Streckenabschnitt (TSC) bei jeweils 10.000 absolvierten Aufträgen im Zeitverlauf zu entnehmen. Es ist jeweils die Entwicklung für die drei Planungsarten Anschlussplanung (AP), ereignisorientiert angestoßene rollierende Planung (RP) und die in dieser Arbeit entwickelte ereignisorientiert angestoßene rollierende Planung mit Verhaltensantizipation und -regelung (hierarchische Planung, HP/RTDP) dokumentiert.

Wie zu erkennen ist, schließen im Verhältnis zur Anschlussplanung die beiden rollierenden Planungsverfahren mit einer Reduzierung im Energieverbrauch, in der Anzahl von Ausfällen und in der Anzahl von Ladezyklen bei der Entwicklung im Zeitverlauf deutlich besser ab. Insb. ist diese Reduzierung bei dem in dieser Arbeit entwickelten Verfahren signifikant höher als die bei der rein rollierenden Verhaltensplanung (vgl. RP und HP/RTDP, z. B. in Abb. 6.12). Der Vergleich von Energieverbrauch und Aufbaubewegungen (vgl. Abb. 6.12 und 5.1.1) zeigt zudem den Einfluss der überlagernden Planungsebene. Die Instruktionen der Top-Ebene beeinflussen die Verhaltensplanung und Ausführung der Basis-Ebene derart, dass

diese Operationsmodi einplant, die aus langfristiger Sicht weniger Energie verbrauchen, aber dabei etwas mehr Aufbaubewegungen, und zwar noch im Rahmen der Nebenbedingung für den Minimalkomfort (vgl. Abs. 6.1.2), zulassen. Dies erklärt die nahezu im gleichen Verhältnis zur Reduzierung des Energieverbrauchs zunehmenden Aufbaubewegungen. Ebenso spiegelt die Konvergenz im Zeitverlauf eine besser werdende Verhaltensantizipation aufgrund des zunehmenden Datenbestands des erweiterten adaptiven MDPs, verursacht durch die Rückkopplung mit dem Systemumfeld, wider.



(a) Energieverbrauch

(b) Aufbaubewegungen

(c) Anzahl Ausfälle

(d) Ladezyklen

Abbildung 6.11: Energieverbrauch $[ws]$, Aufbaubewegungen $[m/s^2]$, #Ausfälle und #Ladezyklen pro Streckenabschnitt (TSC) nach 10.000 absolvierten Aufträgen

Die Abbildung 6.11 zeigt zusammenfassend die Werte zum Energieverbrauch, den Aufbaubewegungen, der Anzahl von Ausfällen sowie der Anzahl von Ladezyklen pro Streckenabschnitt nach der Absolvierung von 10.000 Aufträgen. Die zugehörigen Ergebniswerte zum Experiment I (ein Startwert) sind der Tabelle 6.2 zu entnehmen.

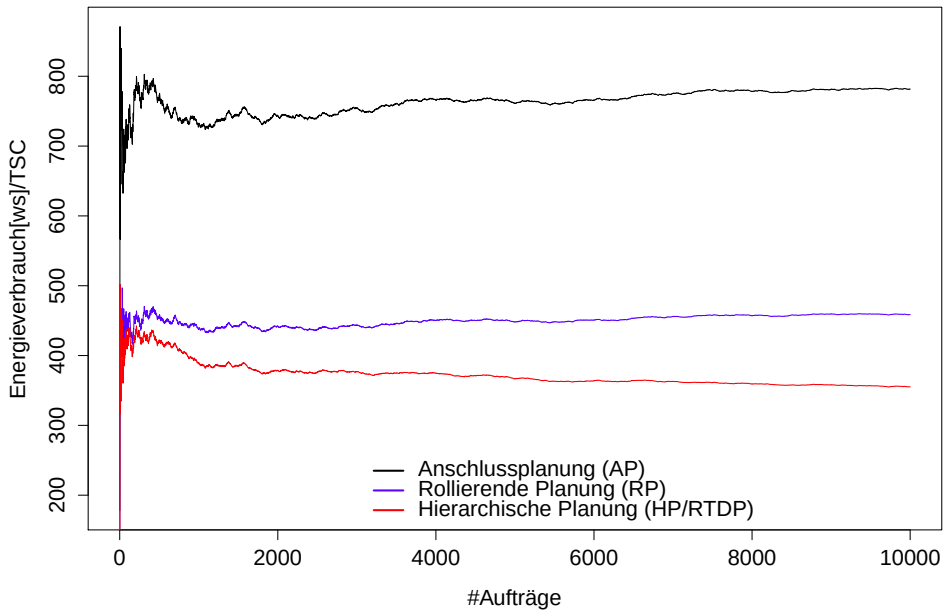


Abbildung 6.12: Entwicklung des durchschnittlichen Energieverbrauchs in $[ws]$ pro Streckenabschnitt (TSC) bei 10.000 absolvierten Aufträgen im Zeitverlauf

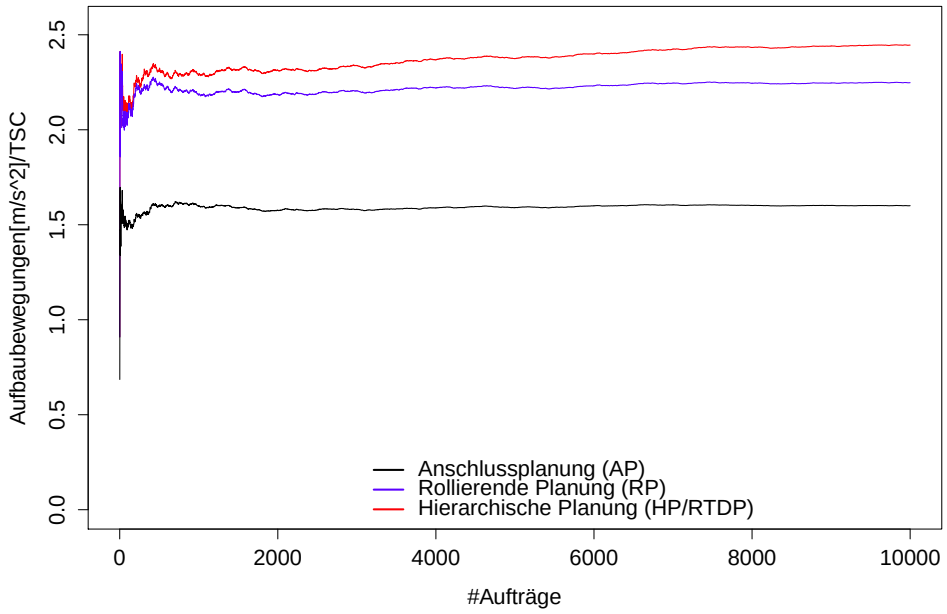


Abbildung 6.13: Entwicklung der durchschnittlichen Aufbaubewegungen in $[m/s^2]$ pro Streckenabschnitt (TSC) bei 10.000 absolvierten Aufträgen im Zeitverlauf

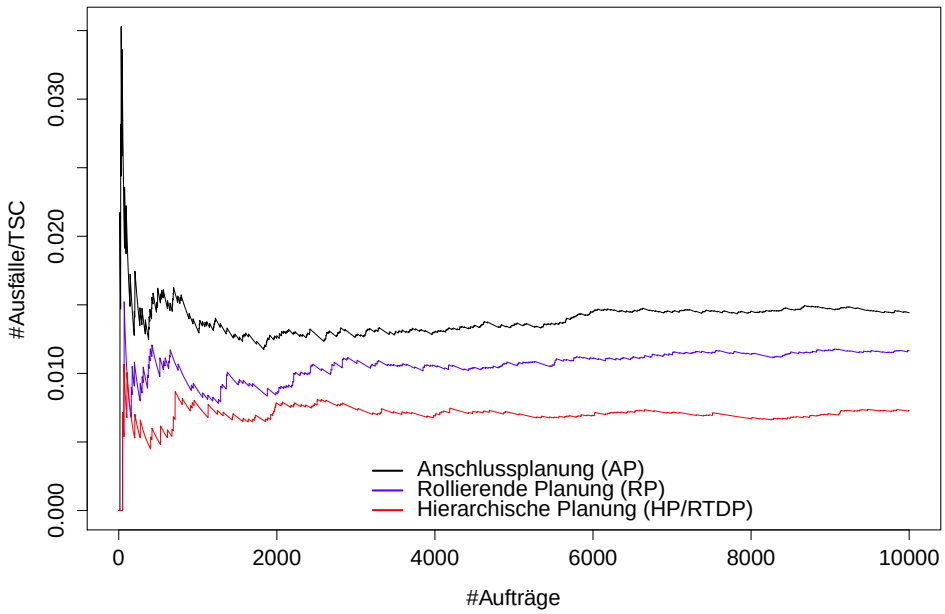


Abbildung 6.14: Entwicklung der durchschnittlichen Anzahl von Ausfällen pro Streckenabschnitt (TSC) bei 10.000 absolvierten Aufträgen im Zeitverlauf

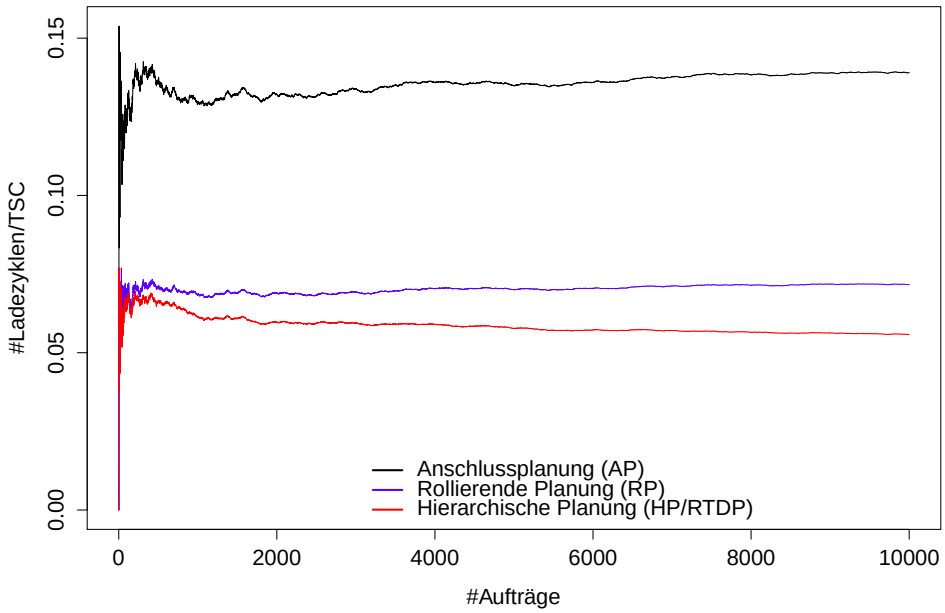


Abbildung 6.15: Entwicklung der durchschnittlichen Anzahl von Ladezyklen pro Streckenabschnitt (TSC) bei 10.000 absolvierten Aufträgen im Zeitverlauf

Stellt man die ereignisorientiert angestoßene rollierende Planung (RP) und die ereignisorientiert angestoßene rollierende Planung mit Verhaltensantizipation und -regelung (HP/RTDP) gegenüber und setzt diese ins Verhältnis zur Anschlussplanung (AP), so kann die jeweilige Verbesserung gezeigt werden. Während die ereignisorientiert angestoßene rollierende Planung (RP) beispielsweise den Energieverbrauch im Verhältnis zur Anschlussplanung (AP) pro Streckenabschnitt (TSC) um $41,320\%$ reduziert, so erlangt die ereignisorientiert angestoßene rollierende Planung mit Verhaltensantizipation und -regelung $54,549\%$ (vgl. Tab. 6.3). Dies entspricht einer weiteren Reduzierung des Energieverbrauchs um $13,229\%$.

	AP	**RP**	**HP/RTDP**
#Aufträge	10.000,000	10.000,000	10.000,000
#Streckenabschnitte (TSC)	27.454,000	27.454,000	27.454,000
#Ausfälle	396,000	320,000	200,000
#Ladezyklen	3.817,000	1.968,000	1.533,000
Aufbaubewegungen $[m/s^2]$ (f1)	43.929,913	61.707,923	67.137,906
Energieverbrauch $[ws]$ (f2)	21.459.738,176	12.592.659,576	9.753.769,706
#Ausfälle pro TSC	0,014	0,012	0,007
#Ladezyklen pro TSC	0,139	0,072	0,056
Aufbaubewegungen $[m/s^2]$ (f1) pro TSC	1,600	2,248	2,445
Energieverbrauch $[ws]$ (f2) pro TSC	781,662	458,682	355,277

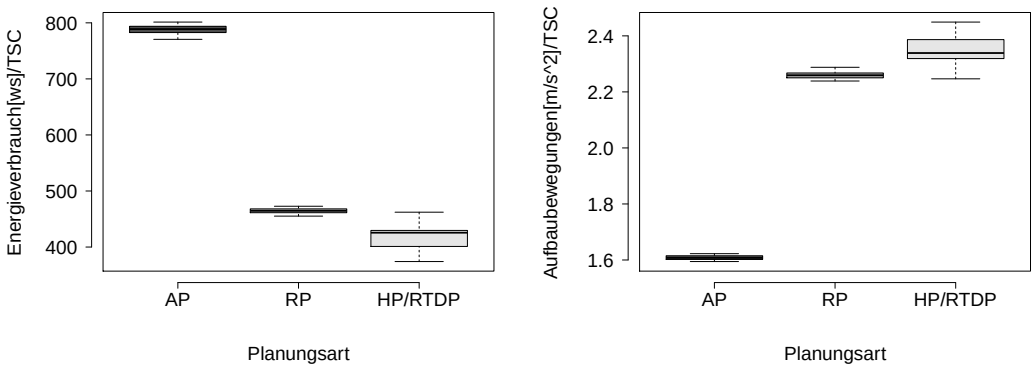
Tabelle 6.2: Übersicht der Ergebniswerte zu Experiment I (ein Startwert)

	RP	**HP/RTDP**
Ausfälle pro TSC	$-19,192\%$	$-49,495\%$
Ladezyklen pro TSC	$-48,441\%$	$-59,838\%$
Aufbaubewegungen pro TSC	$+40,469\%$	$+52,830\%$
Energieverbrauch pro TSC	$-41,320\%$	$-54,549\%$

Tabelle 6.3: Vergleich von rollierender Planung (RP) und hierarchischer Planung (HP/RTDP) im Verhältnis zur Anschlussplanung (AP) auf Basis von Experiment I (ein Startwert)

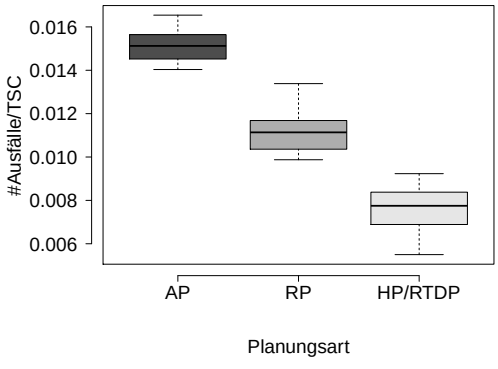
Die Abbildung 6.16 zeigt die korrespondierenden Ergebnisse zum Experiment I von 20 Durchläufen der Simulation zum Beispielmodell mit verschiedenen Startwerten für den Zufallsgenerator zum Generieren der Aufträge und Windverhältnisse. Die zugehörigen durchschnittlichen Ergebniswerte über sämtliche Durchläufe können der Tabelle 6.4 entnommen werden.

Wie der Abbildung 6.16 zu entnehmen ist, können die Verbesserungen durch die ereignisorientiert angestoßene Planung mit Verhaltensantizipation und -regelung (HP/RTDP) auch hierbei bestätigt werden (vgl. Tab. 6.5).

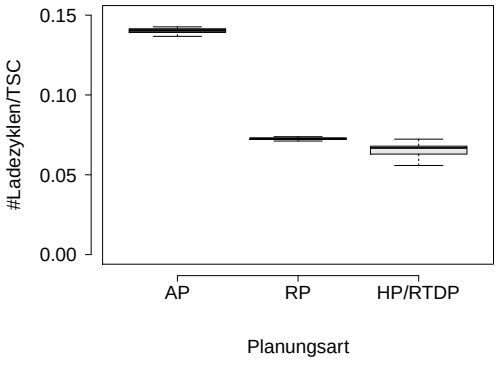


(a) Energieverbrauch

(b) Aufbaubewegungen



(c) Anzahl Ausfälle



(d) Anzahl Ladezyklen

Abbildung 6.16: Energieverbrauch $[ws]$, Aufbaubewegungen $[m/s^2]$, #Ausfälle und #Ladezyklen pro Streckenabschnitt (TSC) nach 10.000 absolvierten Aufträgen im Durchschnitt bei 20 Durchläufen mit verschiedenen Startwerten

	AP	**RP**	**HP/RTDP**
#Aufträge	10.000,000	10.000,000	10.000,000
#Streckenabschnitte (TSC)	27.491,600	27.491,600	27.491,600
#Ausfälle	415,550	305,700	209,000
#Ladezyklen	3.855,650	1.995,300	1.790,000
Aufbaubewegungen $[m/s^2]$ (f1)	44.224,920	62.129,490	64.739,760
Energieverbrauch $[ws]$ (f2)	21.662.418,548	12.768.204,893	11.390.834,112
#Ausfälle pro TSC	0,015	0,011	0,008
#Ladezyklen pro TSC	0,140	0,073	0,065
Aufbaubewegungen $[m/s^2]$ (f1) pro TSC	1,609	2,260	2,355
Energieverbrauch $[ws]$ (f2) pro TSC	787,959	464,437	414,329

Tabelle 6.4: Übersicht der Ergebniswerte (Durchschnitt) zu Experiment I (20 Startwerte)

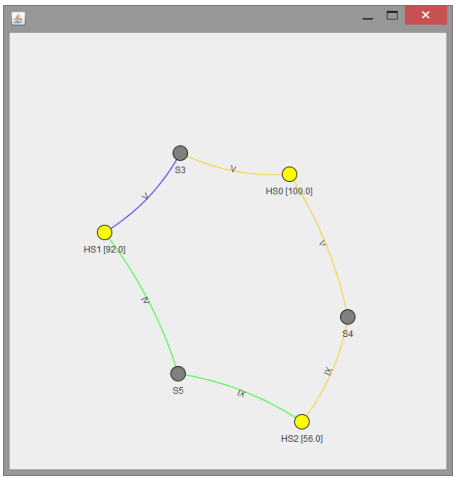
	RP	HP/RTDP
Ausfälle pro TSC	$-26,434\%$	$-49,704\%$
Ladezyklen pro TSC	$-48,250\%$	$-53,575\%$
Aufbaubewegungen pro TSC	$+40,485\%$	$+46,388\%$
Energieverbrauch pro TSC	$-41,058\%$	$-47,417\%$

Tabelle 6.5: Vergleich von rollierender Planung (RP) und hierarchischer Planung (HP/RTDP) im Verhältnis zur Anschlussplanung (AP) auf Basis von Experiment I (20 Startwerte)

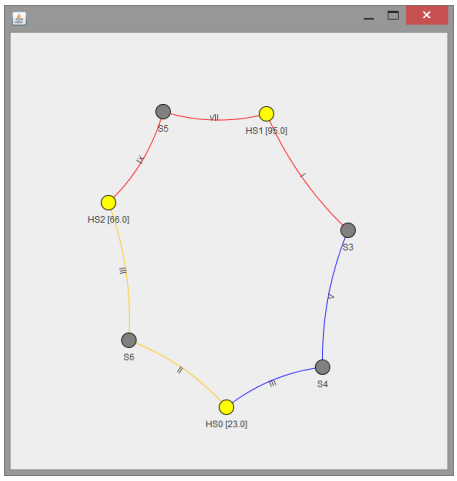
6.2.2 Experiment II: Verhalten in unterschiedlichen Umgebungen

6.2.2.1 Ziel und Beschreibung des Experiments

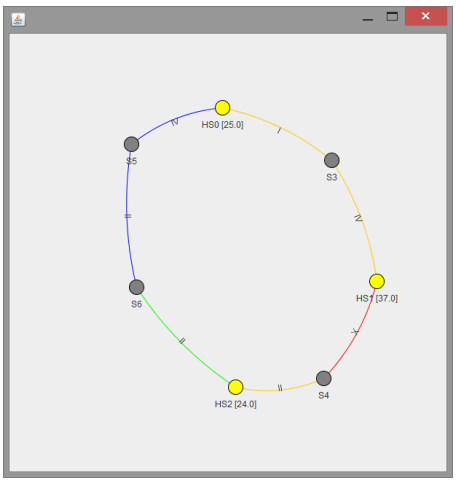
Ziel von Experiment II war es, das Verhalten des RailCabs bei langfristiger Planung und Ausführung anhand des Modells zum aktiven Feder-Neigesystem (vgl. Abs. 6.1.2) im Zusammenschluss mit Modellen von unterschiedlichen Systemumgebungen (vgl. Abs. 6.1.1) zu untersuchen. Damit sollten insb. die Ergebnisse aus Experiment I (vgl. Abs. 6.2.1) modellübergreifend sowie weiterführend bestätigt werden. Zu diesem Zweck wurde ein Modellgenerator entwickelt und damit 16 verschiedene Systemumgebungen für das Experiment II im Vorfeld generiert (vgl. SU01 - SU16 in Abb. 6.17 bis 6.19). Im Anschluss wurde zu jeder Systemumgebung die Planung und Ausführung von 10.000 Aufträgen simuliert und ausgewertet.



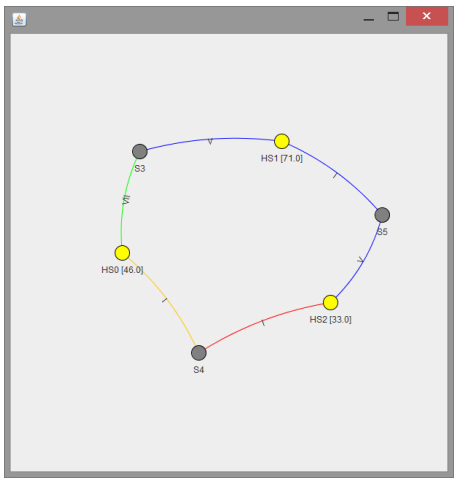
(a) Systemumgebung 1 (SU01)



(b) Systemumgebung 2 (SU02)

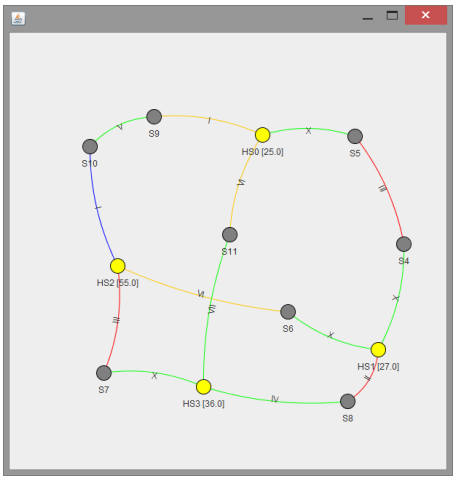


(c) Systemumgebung 3 (SU03)

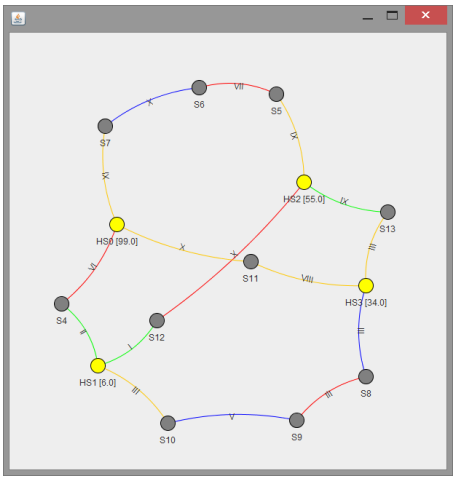


(d) Systemumgebung 4 (SU04)

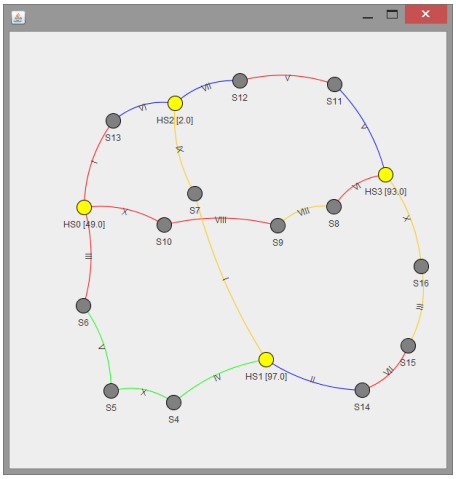
Abbildung 6.17: Exp. II – Zufällig generierte Systemumgebungen SU01 bis SU04



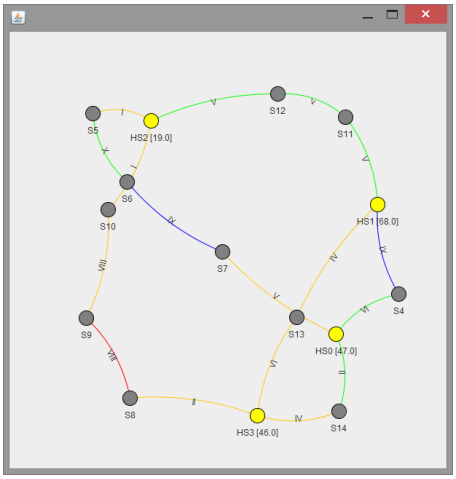
(a) Systemumgebung 5 (SU05)



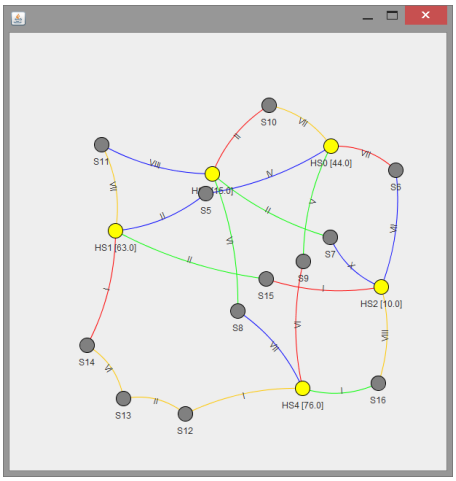
(b) Systemumgebung 6 (SU06)



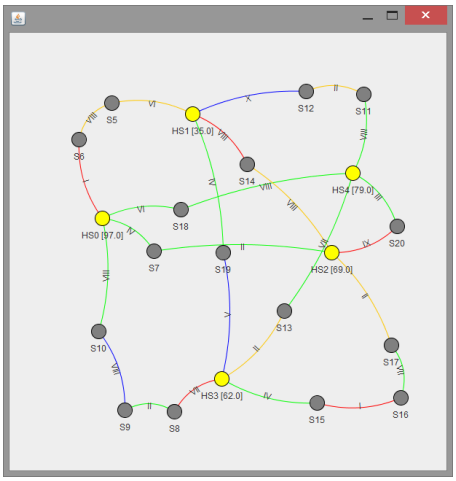
(c) Systemumgebung 7 (SU07)



(d) Systemumgebung 8 (SU08)

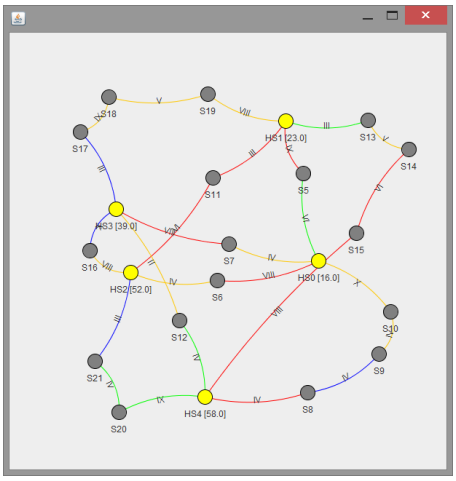


(e) Systemumgebung 9 (SU09)

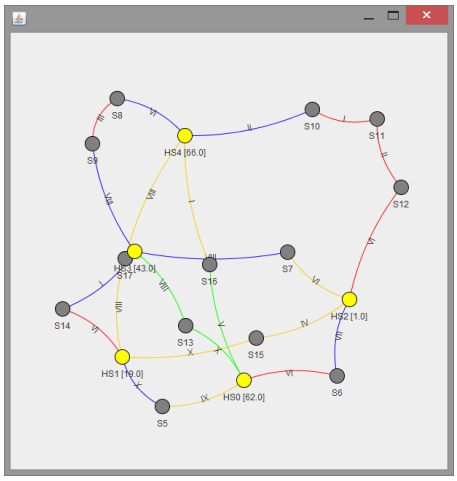


(f) Systemumgebung 10 (SU10)

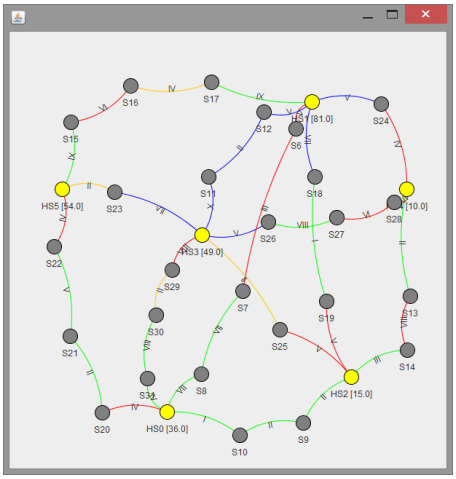
Abbildung 6.18: Exp. II – Zufällig generierte Systemumgebungen SU05 bis SU10



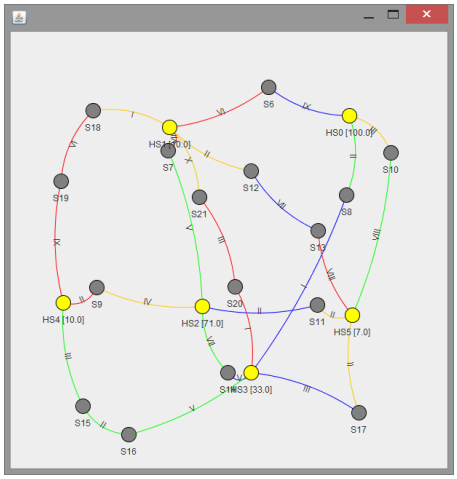
(a) Systemumgebung 11 (SU11)



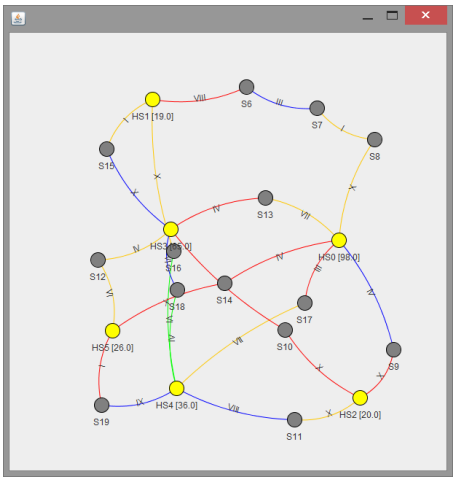
(b) Systemumgebung 12 (SU12)



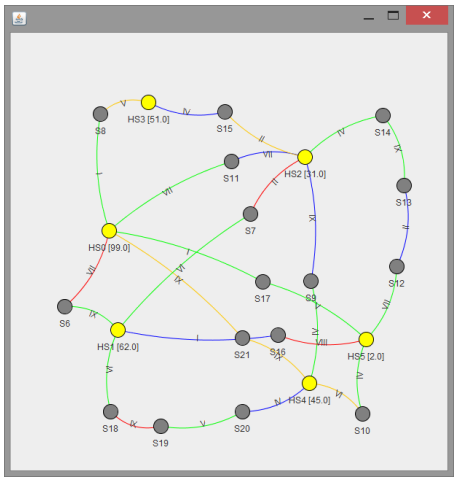
(c) Systemumgebung 13 (SU13)



(d) Systemumgebung 14 (SU14)



(e) Systemumgebung 15 (SU15)



(f) Systemumgebung 16 (SU16)

Abbildung 6.19: Exp. II – Zufällig generierte Systemumgebungen SU11 bis SU16

6.2.2.2 Ergebnisse und Evaluation

Den Abbildungen 6.20 bis 6.23 sind die Ergebnisse der zum Experiment II durchgeführten Simulationen (mehrere Durchläufe mit 16 verschiedenen Systemumgebungen) in Form von Balkendiagrammen über die resultierenden Werte zum Energieverbrauch, zu den Aufbaubewegungen, zur Anzahl von Ausfällen und zur Anzahl von Ladezyklen pro Streckenabschnitt (TSC) bei jeweils 10.000 absolvierten Aufträgen zu entnehmen. Zu jeder Systemumgebung SU01 bis SU16 (vgl. Abs. 6.2.2.1) sind die Ergebnisse für die drei Planungsarten Anschlussplanung (AP), ereignisorientiert angestoßene rollierende Planung (RP) und die in dieser Arbeit entwickelte ereignisorientiert angestoßene rollierende Planung mit Verhaltensantizipation und -regelung (hierarchische Planung, HP/RTDP) abgetragen.

Wie in Abbildung 6.20 zu sehen ist, kann die im Vergleich zur Anschlussplanung (AP) bereits in den Ergebnissen aus Experiment I (vgl. Abs. 6.2.1) zu findende Reduzierung des Energieverbrauchs bei den ereignisorientiert angestoßenen rollierenden Planungsarten (RP und HP/RTDP) für die verschiedenen Systemumgebungen übergreifend bestätigt werden. Wie in den Ergebnissen aus Experiment I, ist diese Reduzierung für die Planungsart mit Verhaltensantizipation und -regelung (HP/RTDP) signifikant höher. Gleichermaßen ist auch ein Anstieg der Aufbaubewegungen im Rahmen der Nebenbedingung zum Mindestkomfort (vgl. Abs. 6.1.2) zu verzeichnen (vgl. Abb. 6.21). Während außerdem die Ladezyklen durchgängig reduziert werden konnten (vgl. Abb. 6.23), ist jedoch bei einigen der Systemumgebungen die Anzahl von Ausfällen bei den ereignisorientiert angestoßenen rollierenden Planungsarten im Vergleich zur Anschlussplanung angestiegen (vgl. z. B. SU04 in Abb. 6.22). Dies ist folgendermaßen zu erklären. Aus Sicht der Basis-Ebene bzw. eines Einzelauftrages ist der Planungshorizont durch den Zielzustand begrenzt. Die rollierende Vorgehensweise führt dazu, dass mit Annäherung zum Ziel immer weniger Streckenabschnitte im Zusammenschluss betrachtet werden. Da die Minimierung der Ausfälle nicht Teil der Zielfunktion ist, wird diesbezüglich die Wahl von Aktionen hinsichtlich der Energiereserve für nachfolgende Aufträge zunehmend riskanter (höheres Ausfallrisiko). Dies erklärt auch, warum dieser Effekt bei der rollierenden Planungsart mit Verhaltensantizipation und -regelung deutlich seltener und mit geringerem Ausmaß stattfindet (vgl. Abb. 6.22). Die Verhaltensantizipation und -regelung bezieht nämlich aufgrund des prädiktiven Modells auftragsübergreifende Informationen mit ein.

Die Tabelle 6.6 fasst die Ergebniswerte zur Verbesserung durch die hierarchische Verhaltensplanung mit Verhaltensantizipation und -regelung (HP/RTDP) im Verhältnis zur Anschlussplanung zusammen und stellt diese denen aus der rollierenden Planung (RP) gegenüber. Insgesamt lässt sich also die eingangs getroffene Annahme, dass die aufgrund des begrenzten Planungshorizontes der Basis-Ebene und der damit einhergehenden Einplanung von aus langfristiger Sicht ggf. suboptimalen Folgen an Operationsmodi zu viel verursachter Energieverbrauch, durch die Instruktionen der übergreifenden Planung der Top-Ebene weiter reduziert bzw. kompensiert werden können, bestätigen.

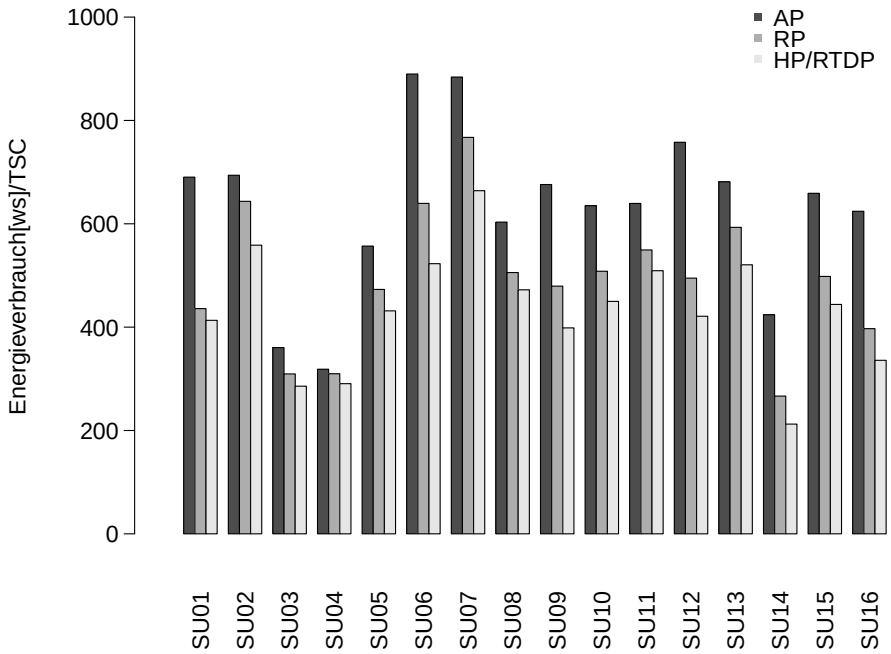


Abbildung 6.20: Durchschnittlicher Energieverbrauch pro Streckenabschnitt (TSC) in $[ws]$ für Systemumgebung 1 bis 16 bei jeweils 10.000 absolvierten Aufträgen

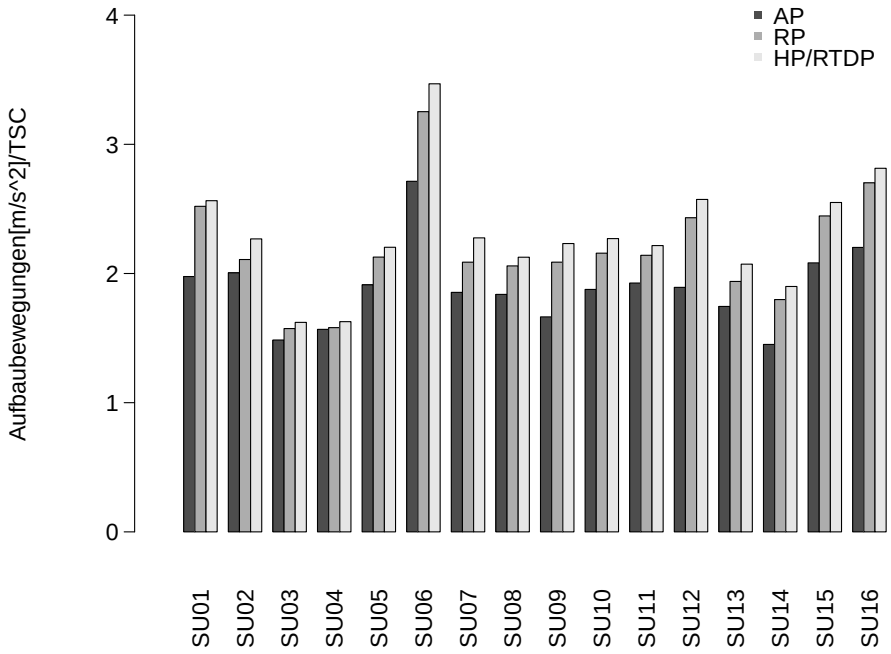


Abbildung 6.21: Durchschnittliche Aufbaubewegungen pro Streckenabschnitt (TSC) in $[m/s^2]$ für Systemumgebung 1 bis 16 bei jeweils 10.000 absolvierten Aufträgen

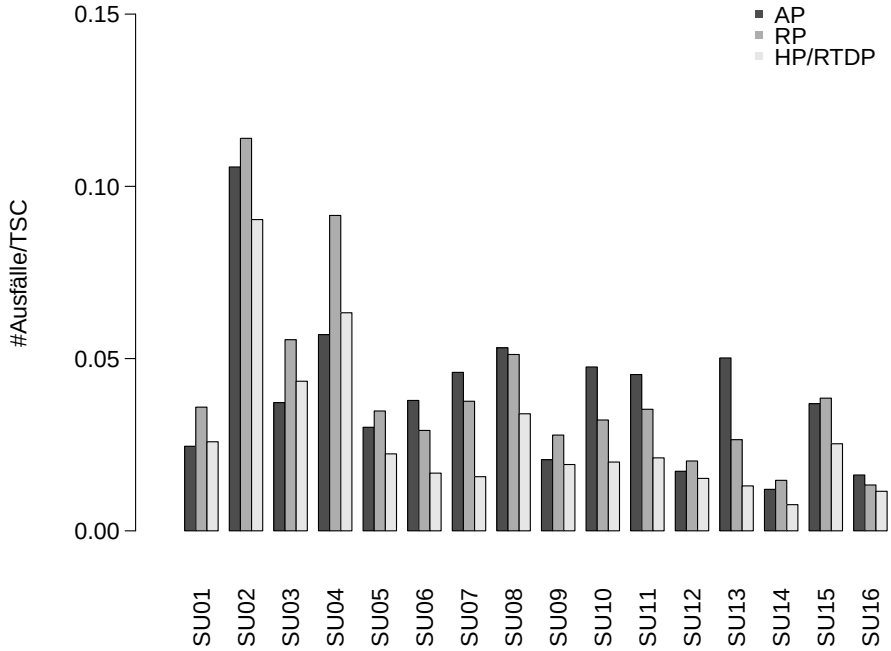


Abbildung 6.22: Durchschnittliche Anzahl von Ausfällen pro Streckenabschnitt (TSC) für Systemumgebung 1 bis 16 bei jeweils 10.000 absolvierten Aufträgen

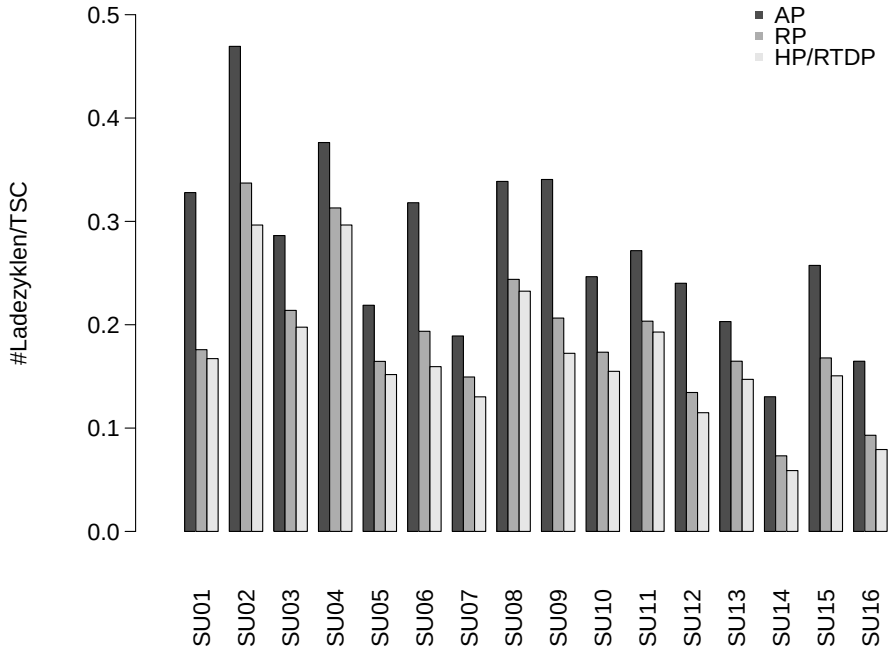


Abbildung 6.23: Durchschnittliche Anzahl von Ladezyklen pro Streckenabschnitt (TSC) für Systemumgebung 1 bis 16 bei jeweils 10.000 absolvierten Aufträgen

SU	Energieverbrauch		Aufbaubewegungen		Ausfälle		Ladezyklen	
	RP	HP/RTDP	RP	HP/RTDP	RP	HP/RTDP	RP	HP/RTDP
01	−36,87%	−40,14%	+27,50%	+29,71%	+46,23%	+5,30%	−46,36%	−49,00%
02	−7,27%	−19,50%	+5,14%	+13,11%	+7,87%	−14,46%	−28,18%	−36,83%
03	−14,13%	−20,72%	+5,96%	+9,19%	+49,04%	+16,67%	−25,29%	−30,96%
04	−2,77%	−8,79%	+0,84%	+3,80%	+60,76%	+11,15%	−16,82%	−21,21%
05	−15,07%	−22,52%	+11,22%	+15,20%	+15,71%	−25,70%	−24,81%	−30,64%
06	−28,14%	−41,25%	+19,86%	+27,81%	−23,01%	−55,75%	−39,11%	−49,87%
07	−13,21%	−24,88%	+12,61%	+22,76%	−18,23%	−65,84%	−20,99%	−31,14%
08	−16,18%	−21,72%	+12,00%	+15,65%	−3,63%	−36,04%	−27,99%	−31,36%
09	−29,09%	−41,04%	+25,45%	+34,11%	+34,62%	−6,78%	−39,38%	−49,39%
10	−19,98%	−29,16%	+14,95%	+20,96%	−32,36%	−58,03%	−29,65%	−37,15%
11	−14,08%	−20,37%	+11,15%	+15,05%	−22,21%	−53,31%	−25,12%	−28,99%
12	−34,69%	−44,43%	+28,46%	+35,95%	+17,23%	−12,01%	−44,02%	−52,16%
13	−12,93%	−23,57%	+11,09%	+18,76%	−47,28%	−74,01%	−18,89%	−27,54%
14	−37,11%	−49,90%	+23,93%	+30,92%	+21,69%	−36,95%	−43,88%	−54,83%
15	−24,41%	−32,63%	+17,45%	+22,45%	+4,34%	−31,55%	−34,78%	−41,50%
16	−36,40%	−46,21%	+22,71%	+27,83%	−17,86%	−29,17%	−43,50%	−51,87%
Ø	−21,39%	−30,43%	+15,65%	+21,45%	+5,81%	−29,16%	−31,80%	−39,03%

Tabelle 6.6: Exp. II – Ergebniswerte zur Verbesserung in 16 verschiedenen Umgebungen bei jeweils 10.000 absolvierten Aufträgen

7 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit bestand in der Umsetzung einer Verhaltensantizipation und -regelung kognitiver mechatronischer Systeme bei langfristiger Planung und Ausführung. Das autonome und zielgerichtete Handeln in dem nichtdeterministischen sowie durch kontinuierliche Prozesse geprägten Umfeld waren dabei von besonderem Interesse. Die Hauptaufgabe bestand darin, eine Planung und Ausführung im Zuge eines langfristigen ökonomischen Betriebs dieser Systeme zu realisieren.

Frühere Arbeiten hatten gezeigt, dass diese Art von Systemen bereits in der Lage ist das erforderliche Verhalten ihrer Funktionsmodule zur Erfüllung von Einzelaufträgen eigenständig und proaktiv zu planen. Dazu wurde eine Sequenz von Aktionen (Operationsmodi des mechatronischen Systems) im Voraus bestimmt, die die bestmögliche Zielerreichung (in der Regel die Minimierung des Ressourcenverbrauchs) erlaubt. Die Echtzeitanforderungen ließen nur eine geringe Zeit zur Durchführung der Verhaltensplanung zu. Daher war der Planungshorizont als kurzfristig einzustufen. Zusätzlich war dieser durch den Zielzustand des aktuellen Einzelauftrages begrenzt und es fehlten Informationen über mögliche Folgeaufträge, die über den Planungshorizont hinaus an das System herangetragen wurden. Folglich konnten die Systeme die Planung im Zuge einer weiterführenden Ausführung von Einzelaufträgen im Vorfeld nicht autonom ausrichten. Hieraus ergab sich die Motivation dieser Arbeit.

Zunächst fand mit Kapitel 2 eine detaillierte Betrachtung der Verhaltensplanung bei begrenztem Planungshorizont statt. Dabei sollte ein grundlegendes Verständnis für das Verhalten, die Ziele und die Verhaltensplanung selbst entwickelt werden. Zudem wurden insb. die aus dem begrenzten Planungshorizont resultierenden Probleme aufgezeigt (vgl. Abs. 2.1). Es folgte die Betrachtung der langfristigen Verhaltensplanung und Ausführung von kognitiven mechatronischen Systemen (vgl. Abs. 2.2). Die Komponenten Verhaltensantizipation, -regelung und -planung wurden voneinander abgrenzt und in das Schema einer hierarchischen Planung eingeordnet. Mit einer Unterscheidung zwischen Top- und Basis-Ebene folgte die Definition der angestrebten Wirkzusammenhänge der Komponenten innerhalb dieser Hierarchie (vgl. Abs. 2.2.1). Es wurde das Ziel verfolgt, Informationen zu charakteristischen Ausführungsverläufen zu sammeln und damit ein prädiktives Modell auf der Top-Ebene zur Verhaltensantizipation der Basis-Ebene aufzubauen. Als Lösungsansatz diente das Erfassen von ausgeführten Plänen während einer ereignisorientiert angestoßenen rollierenden Verhaltensplanung (vgl. Abs. 2.2.2). Schließlich wurden die Anforderungen zur Realisierung des Vorhabens konkretisiert und zu lösende Teilprobleme zerlegt (vgl. Abs. 2.2.3). Für das prädiktive Modell auf der Top-Ebene ergab sich die Anforderung der Verhaltens- sowie Ergebnisorientiertheit (vgl. Abs. 2.2.3.1). Diese

sollte zum einen das Abbilden von Zuständen sowie Aktionen und zum anderen das Bewerten von Auswirkungen selektierter Aktionen während der Verhaltensantizipation ermöglichen. Das Berücksichtigen von (quasi-)kontinuierlichen Zustandswerten des kognitiven mechatronischen Systems und die daraus resultierende hohe Anzahl von unterschiedlichen Zuständen im prädiktiven Modell der Top-Ebene führte zur Anforderung der Klassifikation (vgl. Abs. 2.2.3.2). Diese sollte sowohl adäquate Analysen sowie das Herstellen von Situationsbezügen mittels Klassenbildung als auch eine Klassifizierung neuer zur Betriebszeit zu erfassender Pläne sicherstellen.

Im Anschluss erfolgte in Kapitel 3 die Betrachtung bestehender Arbeiten sowie Techniken. Es wurde untersucht, welchen Beitrag diese zum Lösen der zuvor genannten Teilprobleme leisten können. Zu diesem Zweck wurde zunächst die Anforderung der verhaltens- und ergebnisorientierten Antizipation wieder aufgegriffen sowie entsprechende prädiktive Modelle und Lösungsverfahren untersucht (vgl. Abs. 3.2). Als zielführend erwies sich ein prädiktives Modell auf Grundlage von MDPs und ein Lösungsverfahren basierend auf RTDP (vgl. Abs. 3.2.3). Danach waren Techniken zur Klassifikation Gegenstand näherer Betrachtung (vgl. Abs. 3.3). Es wurde untersucht, wie diese zur Erfüllung der Anforderung von Zustandsklassifikation beitragen können. Als geeignete Grundlage zur Klassenbildung ergab sich ein partitionierendes Clustering auf Basis von K-Means sowie zur Klassifizierung ein Ansatz basierend auf k-Nächste-Nachbar-Klassifikatoren (vgl. Abs. 3.3.3).

Im weiteren Verlauf fand mit der Präzisierung der zu leistenden Arbeit in Kapitel 4 ein Übergang zur Konzeption der Verhaltensantizipation und -regelung für kognitive mechatronische Systeme bei langfristiger Planung und Ausführung in Kapitel 5 statt.

Zuerst wurde die Erfassung und Klassifikation von ausgeführten Plänen entwickelt (vgl. Abs. 5.1). Dazu wurde zunächst ein adaptiver MDP mit Methoden zum Suchen, Hinzufügen und Entfernen von Plänen aufgebaut (vgl. Abs. 5.1.1). Um die Klassenbildung von Zuständen zu realisieren, fand nachfolgend eine Erweiterung des adaptiven MDPs um Zustandsklassen statt (vgl. Abs. 5.1.2). Daran anschließend wurden zur Umsetzung der Klassifizierung von Plänen klassifikationsbasierte Methoden zum Suchen, Hinzufügen und Entfernen von Plänen formuliert (vgl. Abs. 5.1.2). Darauf aufbauend folgte die Konzeption der Verhaltensantizipation (vgl. Abs. 5.2). Hierfür wurde vorbereitend die Kostenbewertung von Plänen entwickelt (Abs. 5.2.1) und später eine RTDP-gestützte verhaltens- und ergebnisorientierte Echtzeit-Antizipation umgesetzt. Schließlich fand die Integration der Verhaltensregelung in die hierarchische Verhaltensplanung statt (vgl. Abs. 5.3).

Darauf folgend wurde in Kapitel 6 die experimentelle Validierung der in Kapitel 5 konzipierten Verfahren mittels Simulation unter Verwendung eines Anwendungsmodells durchgeführt. Zu diesem Zweck wurde für ein repräsentatives mechatronisches System auf das im SFB 614 entwickelte RailCab-System zurückgegriffen. Als konkretes Beispiel für einen Anwendungsfall dienten die langfristige Planung und Ausführung des aktiven Feder-Neigesystems eines einzelnen RailCabs (vgl. Abs. 6.1). Zur anwendungsnahen Simulation des langfristigen Verhaltens wurde zum einen ein Modell der Systemumgebung (vgl. Abs. 6.1.1) und zum anderen ein darin in-

tegriertes Modell des RailCabs am Beispiel des aktiven Feder-Neigesystems (vgl. Abs. 6.1.2) aufgestellt. Abschließend fand die Durchführung von Experimenten zur Validierung der Konzeption statt. Dabei wurde insb. das langfristige Verhalten des Systems, sowohl mit als auch ohne Verhaltensantizipation und -regelung, fokussiert und gegenübergestellt. Schließlich wurden die Ergebnisse zu den durchgeführten Experimenten ausgewertet (vgl. Abs. 6.2).

7.1 Ergebnis

Mithilfe des Modells zum Anwendungsfall RailCab konnte das Verhalten eines repräsentativen kognitiven mechatronischen Systems bei langfristiger Planung und Ausführung von Aufträgen am Beispiel des aktiven Feder-Neigesystems simuliert und damit untersucht werden. Die Ergebnisse der durchgeführten Experimente lieferten Informationen zur langfristigen Entwicklung des durchschnittlichen Energieverbrauchs, den Aufbaubewegungen, der Anzahl von Ausfällen und der Anzahl von Ladezyklen pro Streckenabschnitt. Mit der parallelen Betrachtung von den drei Planungsarten Anschlussplanung, rollierender Planung und der in dieser Arbeit entwickelten hierarchischen Planung mit Verhaltensantizipation und -regelung war ein direkter Vergleich mit bestehenden Verfahren sowie eine Bewertung durchführbar. Zu den in dieser Arbeit generierten Systemumgebungen konnte gezeigt werden, dass bei der Gegenüberstellung von rollierender und hierarchischer Planung im Vergleich zur Anschlussplanung die hierarchische Planung langfristig zu einem ökonomischeren Betrieb des Systems führt. So konnten beispielsweise gegenüber der rollierenden Planung der Energieverbrauch, die Anzahl von Ausfällen und die Anzahl von Ladezyklen signifikant reduziert werden. Der aufgrund des begrenzten Planungshorizontes und der damit einhergehenden Einplanung von aus langfristiger Sicht ggf. suboptimalen Sequenzen an Operationsmodi zu viel verursachte Ressourcenverbrauch kann demzufolge durch eine übergreifende hierarchische Planung mit Verhaltensantizipation und -regelung weiter gesenkt bzw. kompensiert werden.

7.2 Weiterer Forschungsbedarf

Das in dieser Arbeit entwickelte Verfahren ist nur ein erster Schritt zur Realisierung eines autonomen sowie langfristig ökonomischen Betriebs von kognitiven mechatronischen Systemen. Zum einen wurde ausschließlich das Verhalten unter Verwendung von zwei für die Gesamthierarchie repräsentativen Planungsebenen, der zeitlich kürzeren und detaillierenden Basis-Ebene und der zeitlich längeren und inhaltlich vergröbernden sowie überlagernden Top-Ebene, ausreichend untersucht. Anknüpfende Forschungsarbeiten sollten die Untersuchung auf eine Mehrebenenbetrachtung ausweiten und dabei die unterschiedlichen zeitlichen sowie inhaltlichen Planungsaspekte weiter einbeziehen. Eine interessante Aufgabenstellung bestünde z. B. darin, herauszufinden, welches Verbesserungspotential, beispielsweise im Sin-

ne einer Ressourcenreduzierung, jede zusätzliche Planungsebene im Verhältnis mit sich bringt. Zum anderen ist, wie durch den erweiterten adaptiven MDP gezeigt, das prädiktive Modell sowie dessen Datenbestand ausschlaggebend für die Qualität der Verhaltensantizipation und -regelung. Zwar wurde im Rahmen der Konzeption des adaptiven MDPs mit den Methoden zum Hinzufügen und Entfernen von Plänen eine grundsätzliche Kontrollierbarkeit der Größe des prädiktiven Modells geschaffen, dennoch besteht zukünftiger Forschungsbedarf in der Verwaltung des Datenbestands. Der gezielte Einsatz von Verfahren zur Abstraktion und Reduktion kann hier einen entscheidenden Beitrag leisten, indem nicht nur die Größe beherrschbar, sondern zusätzlich das gleiche prädiktive Modell in Form von spezifischen Abstraktionsgraden auf höheren oder niedrigeren gelagerten Planungsebenen nutzbar wird. Die bereits entwickelte Klassifikation bietet hier möglicherweise einen interessanten Ansatzpunkt.

Abschließend lässt sich festhalten, dass die in der vorliegenden Arbeit entwickelte Verhaltensantizipation und -regelung einen grundlegenden Beitrag zur langfristigen Planung und Ausführung von kognitiven mechatronischen Systemen leisten kann.

Literaturverzeichnis

[Ada09] Adamy, J.: *Nichtlineare Regelungen*. Springer-Verlag, 2009

[ADG+09] Adelt, P. ; Donoth, J. ; Geisler, J. ; Henkler, S. ; Kahl, S. ; Klöpper, B. ; Münch, E. ; Oberthür, S. ; Paiz, C. ; Podlogar, H. ; Porrmann, M. ; Radkowski, R. ; Romaus, C. ; Schmidt, A. ; Schulz, B. ; Voecking, H. ; Witkowski, U. ; Witting, K.: *Selbstoptimierende Systeme des Maschinenbaus - Definitionen, Anwendungen, Konzepte*. Band 234. HNI-Verlagsschriftenreihe, Universität Paderborn, 2009

[AEH+11] Adelt, P. ; Esau, N. ; Hölscher, C. ; Kleinjohann, B. ; Kleinjohann, L. ; Krüger, M. ; Zimmer, D.: Hybrid Planning for Self-Optimization in Railbound Mechatronic Systems. (2011), Feb., S. 169–194

[AF07] Abonyi, J. ; Feil, B.: *Cluster Analysis for Data Mining and System Identification*. Birkhäuser Basel, 2007

[Alp08] Alpaydin, E.: *Maschinelles Lernen*. Oldenbourg Wissenschaftsverlag, 2008

[AS92] Alden, J. M. ; Smith, R. L.: Rolling Horizon Procedures in Non-homogeneous Markov Decision Processes. In: *Operations Research* 40 (1992), S. 183–194

[Bak77] Baker, K. R.: An experimental study of the effectiviness of rolling schedules in production planning. In: *Decision Sciences 8* (1977), S. 19–27

[BBS95] Barto, A. G. ; Bradtke, S. J. ; Singh, S. P.: Learning to act using real-time dynamic programming. In: *Artificial Intelligence* 72 (1995), Nr. 1, S. 81–138

[BBW07] Beetz, M. ; Buss, M. ; Wollherr, D.: Cognitive Technical Systems - What Is the Role of Artificial Intelligence? In: Hertzberg, J. (Hrsg.) ; Beetz, M. (Hrsg.) ; Englert, R. (Hrsg.): *KI* Bd. 4667, Springer-Verlag, 2007 (Lecture Notes in Computer Science), S. 19–42

[BDH11] Boutilier, C. ; Dean, T. ; Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. (2011)

[Bel57a] Bellman, R.: A Markovian Decision Process. In: *Indiana Univ. Math. J.* 6 (1957), S. 679–684

[Bel57b] Bellman, R. E.: *Dynamic Programming*. Princeton University Press, 1957

[BEPW08] Backhaus, K. ; Erichson, B. ; Plinke, W. ; Weiber, R.: *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 12., vollständig überarbeitete Auflage. Springer-Verlag, 2008

[BFOS83] Breiman, L. ; Friedman, J. H. ; Olshen, R. A. ; Stone, C. J.: CART: Classification and Regression Trees. (1983)

[BG01] Bonet, B. ; Geffner, H.: Planning and control in artificial intelligence: A unifying perspective. In: *Applied Intelligence* 14 (2001), Nr. 3, S. 237–252

[BG07] Ben-Gal, I.: Bayesian Networks. In: *Encyclopedia of Statistics in Quality & Reliability. Wiley & Sons.* (2007)

[BH76] Baker, F. B. ; Hubert, L. J.: A Graph-Theoretic Approach to Goodness-of-Fit in Complete-Link Hierarchical Clustering. In: *Journal of the American Statistical Association* 71 (1976), Nr. 356, S. 870–878

[BH00] Blai, B. ; Hector, G.: Planning with Incomplete Information as Heuristic Search in Belief Space, AAAI Press, 2000, S. 52–61

[BL98] Blobel, V. ; Lohrmann, E.: *Statistische und numerische Methoden der Datenanalyse*. 1. Auflage. Teubner Verlag, 1998

[BPW10] Bacher, J. ; Pöge, A. ; Wenzig, K.: *Clusteranalyse: Anwendungsorientierte Einführung in Klassifikationsverfahren*. 3., Auflage. Oldenbourg, 2010

[BSG03a] Butz, M. ; Sigaud, O. ; Gérard, P.: Internal Models and Anticipations in Adaptive Learning Systems. In: *ABiALS*, 2003, S. 86–109

[BSG03b] Butz, M. ; Sigaud, O. ; Gérard, P.: Anticipatory Behavior: Exploiting Knowledge About the Future to Improve Current Behavior. In: Butz, M. (Hrsg.) ; Sigaud, O. (Hrsg.) ; Gérard, P. (Hrsg.): *Anticipatory Behavior in Adaptive Learning Systems* Bd. 2684. Springer-Verlag, 2003, S. 1–10

[BV08] Bankhofer, U. ; Vogel, J.: Entscheidungsbäume. In: *Datenanalyse und Statistik*. Gabler, 2008, S. 273–284

[CL91] Covrigaru, A. A. ; Lindsay, R. K.: Deterministic Autonomous Systems. In: *AI Magazine* 2(3) (1991), S. 110–117

[Czi08] Czichos, H.: *Mechatronik - Grundlagen und Anwendungen technischer Systeme.* 2., aktualisierte und erweiterte Auflage. Vieweg+Teubner, 2008

[DAE94] Davidsson, P. ; Astor, E. ; Ekdahl, B.: A framework for autonomous agents based on the concept of anticipatory systems. In: *Cybernetics and Systems* 94 (1994), S. 1427–1434

[Dan09] Dangelmaier, W.: *Theorie der Produktionsplanung und -steuerung - Im Sommer keine Kirschpralinen?* Springer-Verlag, 2009

[DIN94] DIN: *DIN 19 226 Teil 1: Leittechnik - Regelungstechnik und Steuerungstechnik - Allgemeine Grundbegriffe.* Deutsche Norm, 1994

[DLR77] Dempster, A. P. ; Laird, N. M. ; Rubin, D. B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1977), Nr. 1, S. 1–38

[DP04] Dittmar, R. ; Pfeiffer, B.-M.: *Modellbasierte prädiktive Regelung: Eine Einführung für Ingenieure.* Oldenbourg Verlagsgruppe, 2004

[DP06] Dittmar, R. ; Pfeiffer, B.-M.: Modellbasierte prädiktive Regelung in der industriellen Praxis (Industrial Application of Model Predictive Control). In: *Automatisierungstechnik* 54 (2006), Nr. 12, S. 590–601

[Dub87] Dubes, R. C.: How many clusters are best? - An experiment. In: *Pattern Recognition* 20 (1987), Nr. 6, S. 645–663

[Dub03] Dubois, D.: Mathematical Foundations of Discrete and Functional Systems with Strong and Weak Anticipations. In: Butz, M. (Hrsg.) ; Sigaud, O. (Hrsg.) ; Gérard, P. (Hrsg.): *Anticipatory Behavior in Adaptive Learning Systems* Bd. 2684. Springer-Verlag, 2003, S. 110–132

[Dum10] Dumitrescu, R.: *Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme*, Universität Paderborn, Dissertation, 2010

[ES00] Ester, M. ; Sander, J.: *Knowledge Discovery in Databases: Techniken und Anwendungen.* Springer-Verlag, 2000

[FGK+04] Frank, U. ; Giese, H. ; Klein, F. ; Oberschelp, O. ; Schmidt, A. ; Schulz, B. ; Voecking, H. ; Witting, K.: *Selbstoptimierende Systeme des Maschinenbaus - Definitionen und Konzepte.* Band 155. Paderborn : HNI-Verlagsschriftenreihe, Universität Paderborn, 2004

[FMW05] Fleischmann, B. ; Meyr, H. ; Wagner, M.: Advanced Planning. In: *Supply Chain Management and Advanced Planning - Concepts, Models Software and Case Studies* (2005), S. 81–106

[Fre86] French, S.: *Decision theory.* Chichester : Horwood [u.a.], 1986 (Ellis Horwood series in mathematics and its applications. Statistics and operational research)

[GD92] Gowda, K. C. ; Diday, E.: Symbolic clustering using a new similarity measure. In: *Systems, Man and Cybernetics, IEEE Transactions on* 22 (1992), mar/apr, Nr. 2, S. 368–378

[Geb09] Gebhard, M.: *Hierarchische Produktionsplanung bei Unsicherheit.* Gabler, 2009

[GNT04] Ghallab, M. ; Nau, D. ; Traverso, P.: *Automated Planning - Theory and Practice.* Elsevier, 2004

[GR69] Gower, J. C. ; Ross, G. J. S.: Minimum Spanning Trees and Single Linkage Cluster Analysis. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 18 (1969), Nr. 1

[Gör95] Görz, G.: *Einführung in die künstliche Intelligenz.* Addison-Wesley, 1995

[Gro00] Groth, R.: *Data Mining - Building Competitive Advantage.* Prentice Hall, 2000

[GRS03] Görz, G. ; Rollinger, C.-R. ; Schneeberger, J.: *Handbuch der Künstlichen Intelligenz.* 4. Auflage. Oldenbourg Wissenschaftsverlag, 2003

[GRS08] Gausemeier, J. (Hrsg.) ; Rammig, F. J. (Hrsg.) ; Schäfer, W. (Hrsg.): *HNI-Verlagsschriftenreihe, Paderborn.* Bd. 223: *Self-optimizing Mechatronic Systems: Design the Future.* Heinz Nixdorf Institut, 2008

[GWTD08] Geisler, J. ; Witting, K. ; Trächtler, A. ; Dellnitz, M.: Multiobjective Optimization of Control Trajectories for the Guidance of a Rail-bound Vehicle. In: *17th World Congress, The International Federation of Automatic Control (IFAC)* (2008), jul

[HK06] Han, J. ; Kamber, M.: *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, 2006

[How60] Howard, Ronald A.: *Dynamic Programming and Markov Processes.* MIT Press, 1960

[Ise08] Isermann, R.: *Mechatronische Systeme - Grundlagen.* 2. vollständig neu bearbeitete Auflage. Springer-Verlag, 2008

[Jae97] Jaeger, H.: A Short Introduction To Observable Operator Models Of Discrete Stochastic Processes. (1997)

[Jae00] Jaeger, H.: Observable operator processes and conditioned continuation representations. In: *Neural computation* 12 (2000), Nr. 6, S. 1371–1398

[Jai10] Jain, A. K.: Data clustering: 50 years beyond K-means. In: *Pattern Recognition Letters* 31 (2010), Nr. 8, S. 651–666

[JD88] Jain, A. K. ; Dubes, R. C.: *Algorithms for Clustering Data*. Prentice Hall College Div, 1988

[JMF99] Jain, A. K. ; Murty, M. N. ; Flynn, P. J.: Data clustering: a review. In: *ACM Comput. Surv.* 31 (1999), September, Nr. 3, S. 264–323

[JS04] James, M. R. ; Singh, S: Learning and discovery of predictive state representations in dynamical systems with reset. In: *In ICML*, 2004, S. 417–424

[JSL04] James, M. R. ; Singh, S. ; Littman, M. L.: Planning with predictive state representations. In: *The 2004 International Conference on Machine Learning and Applications*, 2004

[KAA12] Klöpper, B. ; Aufenanger, M. ; Adelt, P.: Planning for mechatronics systems - Architechture, methods and case study. In: *Engineering Applications of Artificial Intelligence.* 25 Issue 1 (2012), S. 174–188

[Kim98] Kimms, A.: Stability measures for rolling schedules with applications to capacity expansion planning, master production scheduling, and lot sizing. In: *Omega 26* (1998), S. 355–366

[Kin67] King, B.: Step-Wise Clustering Procedures. In: *Journal of the American Statistical Association* 62 (1967), Nr. 317, S. 86–101

[Klö09] Klöpper, B.: *Ein Beitrag zur Verhaltensplanung für interagierende intelligente mechatronische Systeme in nicht-deterministischen Umgebungen*, Fakultät für Wirtschaftswissenschaften, Universität Paderborn, Dissertation, 2009

[KLC98] Kaelbling, L. P. ; Littman, M. L. ; Cassandra, A. R.: Planning and acting in partially observable stochastic domains. In: *Artificial Intelligence* 101 (1998), Nr. 1-2, S. 99–134

[Kor90] Korf, R. E.: Real-time heuristic search. In: *Artificial Intelligence* 42 (1990), Nr. 2-3, S. 189–211

[KR76] Keeney, R. L. ; Raiffa, H.: *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York : John Wiley & Sons, Inc, 1976 (Wiley series in probability and mathematical statistics)

[KR93] Keeney, R. L. ; Raiffa, H.: *Decisions with Multiple Objectives. Preferences and Value Tradeoffs.* Camebridge University Press, 1993

[KRS+08] Klöpper, B. ; Romaus, C. ; Schmidt, A. ; Voecking, H. ; Donoth, J.: Defining Plan Metrics for Multi-Agent Planning Within Mechatronic Systems. In: *Proceedings of IDETC/CIE 2008 ASME 2008 International Design Engineering Technical Conference & Computers and Information in Engineering Conference.* August 3-6, New York, USA, 2008

[KRSV08] Klöpper, B. ; Romaus, C. ; Schmidt, A. ; Voecking, H.: A Multi-Agent Planning Problem for the Coordination of Function Modules. In: Gausemeier, J. (Hrsg.) ; Rammig, F.-J. (Hrsg.) ; Schäfer, W. (Hrsg.): *Self-optimizing Mechatronic Systems: Design the Future*, Heinz Nixdorf Institut, Feb. 2008 (HNI-Verlagsschriftenreihe, Paderborn), S. 377–393

[KSWR12] Klöpper, B. ; Sondermann-Wölke, C. ; Romaus, C.: Probabilistic Planning for Predictive Condition Monitoring and Adaptation within the Self-Optimizing Energy Management of an Autonomous Railway Vehicle. In: *Journal for Robotics and Mechatronics* 24 No.1 (2012), S. 5–15

[KWZ98] Krahl, D. ; Windheuser, U. ; Zick, F.-K.: *Data Mining - Einsatz in der Praxis.* Addison-Wesley, 1998

[LDK95] Littman, M. L. ; Dean, T. L. ; Kaelbling, L. P.: On the complexity of solving Markov decision problems. In: *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence* Morgan Kaufmann Publishers Inc., 1995, S. 394–402

[Lex96] Lexikonredaktion, Meyers: *DUDEN - Das Neue Lexikon.* 3. Auflage. Dudenverlag, 1996

[Lip06] Lippe, W.-M.: *Soft-Computing mit Neuronalen Netzen, Fuzzy-Logic und Evolutionären Algorithmen.* Springer-Verlag, 2006

[LSS02] Littman, M. L. ; Sutton, R. S. ; Singh, S.: Predictive Representations of State. In: *In Advances In Neural Information Processing Systems 14*, MIT Press, 2002, S. 1555–1561

[Mah36] Mahalanobis, P. C.: On the generalised distance in statistics. In: *Proceedings National Institute of Science, India* Bd. 2, 1936, S. 49–55

[Mit97] Mitchell, T. M.: *Machine learning.* 1. McGraw-Hill Science / Engineering / Math, 1997

[Nau00] Naumann, R.: *Modellierung und Verarbeitung vernetzter intelligenter mechatronischer Systeme.* Dissertation, Fakultät Maschinenbau, Universität Paderborn, 2000

[PBCF08] Pezzulo, G. ; Butz, M. ; Castelfranchi, C. ; Falcone, R.: *The Challenge of Anticipation - A Unifying Framework for the Analysis and Design of Artificial Cognitive Systems.* Springer-Verlag, 2008

[PBFG05] Pahl, G. ; Beitz, W. ; Feldhusen, J. ; Grote, K.-H.: *Konstruktionslehre. Grundlagen erfolgreicher Produktentwicklung - Methoden und Anwendung.* 6. Auflage. Springer-Verlag, 2005

[Pet05] Petersohn, H.: *Data Mining. Verfahren, Prozesse, Anwendungsarchitektur.* Oldenbourg Verlag München Wien, 2005

[Pez07] Pezzulo, G.: Anticipation and Future-Oriented Capabilities in Natural and Artificial Cognition. In: Lungarella, Max (Hrsg.) ; Iida, Fumiya (Hrsg.) ; Bongard, Josh (Hrsg.) ; Pfeifer, Rolf (Hrsg.): *50 Years of Artificial Intelligence* Bd. 4850. Springer-Verlag, 2007, S. 257–270

[Poo11] Pook, S.: *Eine Methode zum Entwurf von Zielsystemen selbstoptimierender mechatronischer Systeme*, Universität Paderborn, Dissertation, 2011

[PS78] Puterman, M. L. ; Shin, M. C.: Modified policy iteration algorithms for discounted Markov decision problems. In: *Management Science* 24 (1978), Nr. 11, S. 1127–1137

[Put05] Puterman, M. L.: *Markov decision processes: discrete stochastic dynamic programming.* Wiley-Interscience, 2005

[Qui86] Quinlan, J. R.: Induction of Decision Trees. In: *Mach. Learn.* 1 (1986), März, Nr. 1, S. 81–106

[Qui93] Quinlan, J. R.: *C4. 5: programs for machine learning.* Bd. 1. Morgan Kaufmann, 1993

[RN10] Russell, S. ; Norvig, P.: *Artificial Intelligence - A Modern Approach.* Third Edition. Pearson Education, 2010

[Rod06] Roddeck, W.: *Einführung in die Mechatronik.* 3., überarbeitete und ergänzte Auflage. Teubner Verlag, Wiesbaden, 2006

[Rom94] Rommelfanger, H.: *Fuzzy Decision Support-Systeme - Entscheidung bei Unschärfe.* 2. Auflage. Springer-Verlag, 1994

[Rop79] Ropohl, G.: *Eine Systemtheorie der Technik - Zur Grundlegung der Allgemeinen Technologie.* Carl Hanser Verlag München Wien, 1979

[Rop09] Ropohl, G.: *Allgemeine Technologie - Eine Systemtheorie der Technik.* 3., überarbeitete Auflage. Universitätsverlag Karlsruhe, 2009

[Ros12] Rosen, R.: *Anticipatory Systems - Philosophical, Mathematical, and Methodological Foundations.* Second Edition. Springer-Verlag, 2012

[RS04] Rudary, M. R. ; Singh, S.: A nonlinear predictive state representation. In: *Advances In Neural Information Processing Systems (NIPS)*, 2004

[Sch89] Schweitzer, G.: *Mechatronik - Aufgaben und Lösungen: VDI-Berichte Nr. 787, S. 1/15.* VDI Verlag Düsseldorf, 1989

[Sch94] Schneeweiss, C.: Elemente einer Theorie hierarchischer Planung. In: *Operations-Research-Spektrum* 16 (1994), Nr. 2, S. 161–168

[Sch01] Scholl, A.: *Robuste Planung und Optimierung. Grundlagen - Konzepte und Methoden - Experimentelle Untersuchungen.* Physica-Verlag, 2001

[Sch06] Schlautmann, P.: *Entwicklung eines neuartigen dreidimensionalen aktiven Federungssystems für ein Schienenfahrzeug*, Fakultät für Maschinenbau, Universität Paderborn, Dissertation, 2006

[SFB04] *SFB614: Sonderforschungsbereich 614: Selbstoptimierende Systeme des Maschinenbaus - Finanzierungsantrag.* Paderborn : Universität Paderborn, 2004

[SGDS09] Sanner, S. ; Goetschalckx, R. ; Driessens, K. ; Shani, G.: Bayesian Real-Time Dynamic Programming. In: *IJCAI*, 2009, S. 1784–1789

[SJR04] Singh, S. ; James, M. R. ; Rudary, M. R.: Predictive state representations: a new theory for modeling dynamical systems. In: *Proceedings of the 20th conference on Uncertainty in artificial intelligence.* Arlington, Virginia, United States : AUAI Press, 2004 (UAI '04), S. 512–519

[SLJ+03] Singh, S. ; Littman, M. L. ; Jong, N. K. ; Pardoe, D. ; Stone, P.: Learning Predictive State Representations. In: *In The Twentieth International Conference on Machine Learning (ICML-2003)*, AAAI Press, 2003, S. 712–719

[SS73] Sneath, P. H. A. ; Sokal, R. R.: *Numerical taxonomy - the principles and practice of numerical classification.* Freeman, 1973

[Tho06] Thomanek, H.-J.: Grundlagen der Informationstechnik. In: Altendorfer, Otto (Hrsg.) ; Hilmer, Ludwig (Hrsg.): *Medienmanagement.* VS Verlag für Sozialwissenschaften, 2006, S. 175–206

[TSK06] Tan, P.-N. ; Steinbach, M. ; Kumar, V.: *Introduction to Data Mining.* Addison Wesley, 2006

[Unb08] Unbehauen, H.: Grundlagen der Fuzzy-Regelung. In: *Regelungstechnik I.* Vieweg+Teubner, 2008, S. 337–368

[VDI00] VDI: *VDI-Richtlinie 3780 - Technikbewertung Begriffe und Grundlagen.* Verein Deutscher Ingenieure, 2000

[VDI04] VDI: *VDI-Richtlinie 2206 - Entwicklungsmethodik für mechatronische Systeme.* Verein Deutscher Ingenieure, 2004

[VT08] Vöcking, H. ; Trächtler, A.: Self-optimization of an active suspension system regarding energy requirements. In: *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, 2008, S. 569–574

[Whi93] White, D. J.: A Survey of Applications of Markov Decision-Processes. In: *Journal of the Operational Research Society* 44 (1993), Nr. 11, S. 1073–1096

[Wie07] Wiewiora, E. W.: *Modeling probability distributions with predictive state representations.* La Jolla, CA, USA, Diss., 2007

[Wil91] Wilson, S. W.: *The Animat Path to AI.* 1991

[Ye03] Ye, N.: *The handbook of data mining.* Lawrence Erlbaum, 2003

[Zad65] Zadeh, L. A.: Fuzzy Sets. In: *Information and Control* (1965), Nr. 3, S. 338–353

[Zah71] Zahn, C. T.: Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. In: *Computers, IEEE Transactions on* C-20 (1971), jan., Nr. 1, S. 68–86

[Zil96] Zilberstein, S.: Using Anytime Algorithms in Intelligent Systems. In: *AI Magazine* 17 (1996), Nr. 3, S. 73–83

[ZL96] Zhang, N. L. ; Liu, W.: Planning in stochastic domains: Problem characteristics and approximation. 1996. – Forschungsbericht

[ZLZ13] Zhang, N. L. ; Lee, S. S. ; Zhang, W.: A Method for Speeding Up Value Iteration in Partially Observable Markov Decision Processes. (2013)